Sistemas Periciais e Robótica

REALIZADO POR:
António Pedro Almeida Viegas Alves
César Francisco Gonçalves Rodrigues

Objectivos

Com este trabalho pretende-se criar um programa que facilite os seus futuros utilizadores na difícil tarefa de marcação de reuniões. Irá ser necessário coordenar as várias agendas correspondentes a cada um dos utilizadores de modo a que se possa marcar uma data para uma reunião. Esta data deverá satisfazer a totalidade ou a maioria dos intervenientes na mesma.

Um utilizador é representado por um agente, estando este responsável por comunicar com os restantes agentes que personificam cada um dos utilizadores intervenientes na reunião a marcar. Assim, é possível obter as suas disponibilidades e, a partir destes dados marcar a reunião pedida. A comunicação entre os vários agentes será desenvolvida usando as funcionalidades de uma ferramenta – JATLite (http://java.stanford.edu/) – que facilita a implementação das mesmas.

Pretende-se que esta aplicação possa ser usada por vários utilizadores que partilhem uma rede, poderá ser uma LAN de uma empresa, mas também poderá ser de um modo mais global a própria Internet. Isto permite a qualquer utilizador, que tenha acesso à rede global, marcar reuniões ou encontros com outros utilizadores que se encontrem ligados à mesma.

Motivação

Na exposição dos diversos trabalhos propostos para esta cadeira, este em particular despertou desde logo o nosso interesse por várias razões.

Do ponto de vista académico a implementação de um sistema multi-agente permite-nos ter uma percepção das dificuldades que existe em transpor um programa deste tipo da teoria para a prática. Espera-se que a maior ou menor facilidade com que estes obstáculos são ultrapassados nos dê uma boa percepção da dinâmica de um sistema desta natureza.

Por outro lado, ficaremos a conhecer o processo de comunicação inter-agentes através de linguagens como o KQML.

Fora do âmbito meramente académico as motivações deste trabalho aumentam. A marcação automática de reuniões é um campo ainda pouco desenvolvido, podendo-se assim ter uma boa margem de manobra.

Com o crescimento das organizações torna-se cada vez mais difícil conciliar a marcação de reuniões com todos os intervenientes nestas minimizando as desmarcações de outros eventos. Além disso, o aumento da planificação do trabalho nas empresas verificado actualmente, exige um aumento do número de reuniões.

Se não nos restringirmos o domínio desta aplicação a uma rede local é possível que este seja alargado à Internet. Com a facilidade que hoje em dia existe em obter uma ligação (PDA/WAP/telemóveis de 3ª geração) poder-se-á em qualquer lugar e em qualquer altura consultar a agenda pessoal e marcar novas reuniões. Além disso, é também possível avisar os utilizadores de novas marcações em tempo real através da utilização do mail e SMS.

Ainda é de referir que em certos trabalhos, noutras disciplinas, já existe a necessidade de marcarmos horários para reuniões com os nossos colegas. Esta tarefa torna-se um pouco complexa pois, muitas vezes, é impossível satisfazer todos os intervenientes.

Descrição

Funcionalidades

O trabalho realizado apresenta as seguintes funcionalidades:

- Sistema de segurança

A aplicação criada contém um sistema de segurança que permite identificar o utilizador em causa evitando abusos de privacidade. Assim, para entrar no programa é necessária a introdução de um *login* e de uma *password*.

- Gestão de Agendas

Cada utilizador pode fazer a gestão da sua agenda através de operações de inserção, remoção e alteração de ocupações. Uma ocupação é um intervalo de tempo em que o utilizador tem algo programado, associado a um nível de prioridade.

As agendas são persistentes estando gravadas num ficheiro e são visualizadas por semana. A interacção com a agenda é feita pelo meio de uma janela. Com a ajuda do rato pode-se realizar todas as operações atrás descritas através de selecções directas no horário.

- Marcação de Reuniões

Com a ajuda da aplicação é possível marcar reuniões com os outros utilizadores. Existe a opção de escolher o dia, a duração, a hora pretendida e as pessoas com quem se quer marcar a reunião. A hora da marcação só será a pretendida se satisfizer a média máxima de indisponibilidade. A média máxima de indisponibilidade é a média das prioridades das ocupações de todos os utilizadores, intervenientes na reunião, a uma determinada hora. Este valor é indicado pelo utilizador.

Quando uma marcação é feita todos os utilizadores que entram na reunião são notificados.

- Créditos

É possível consultar os nomes dos autores do programa, respectiva data de criação e versão.

Estrutura do programa

Para este trabalho consideramos quatro módulos principais: a manipulação do horário, o esquema de negociação, as comunicações e a interface gráfica.

A manipulação do horário está directamente relacionada com a gestão da agenda electrónica. Este módulo será responsável pela operações de inserção, remoção, actualização e gravação das ocupações no horário.

Com o módulo da negociação pretende-se estabelecer um conjunto de regras que permitam determinar qual o melhor horário para um determinado número de pessoas.

Uma ocupação poderá encontrar-se em vários estados de acordo com o horário estabelecido pelo utilizador. Estes estados poderão classificar-se de 1 a 5 conforme o grau de prioridade.

Para que o algoritmo implementado funcione é necessário especificar o valor médio máximo de indisponibilidade total. Como já foi referido anteriormente, este é a média das prioridades dos utilizadores a uma determinada hora. É este valor que vai ser a condição de paragem do algoritmo. Se a hora a examinar tiver uma indisponibilidade média total inferior ao especificado é encontrada uma solução.

Quanto à parte das comunicações pensou-se em vários tipos de implementação acabando-se por implementar um esquema de ligação com a ajuda de um router. No esquema escolhido sobressai o facto de por cada agenda existirem dois processos independentes que interagem de modo a formar um agente.

Um trata do interface e em princípio não irá estar a correr na maior parte do tempo. Serve para interagir com o utilizador e tem como vantagens poder encontrar-se em qualquer local e poder aceder a horários de pessoas diferentes a partir do mesmo programa (tendo o login e password dessas pessoas).

O outro é um processo mais fixo que em geral destina-se a correr em background numa máquina que esteja sempre ligada à internet (exemplo: na máquina TOM da faculdade de engenharia). Todo o processo de gestão de alterações no ficheiro que guarda o horário é feito neste.

Processo gestor do interface:

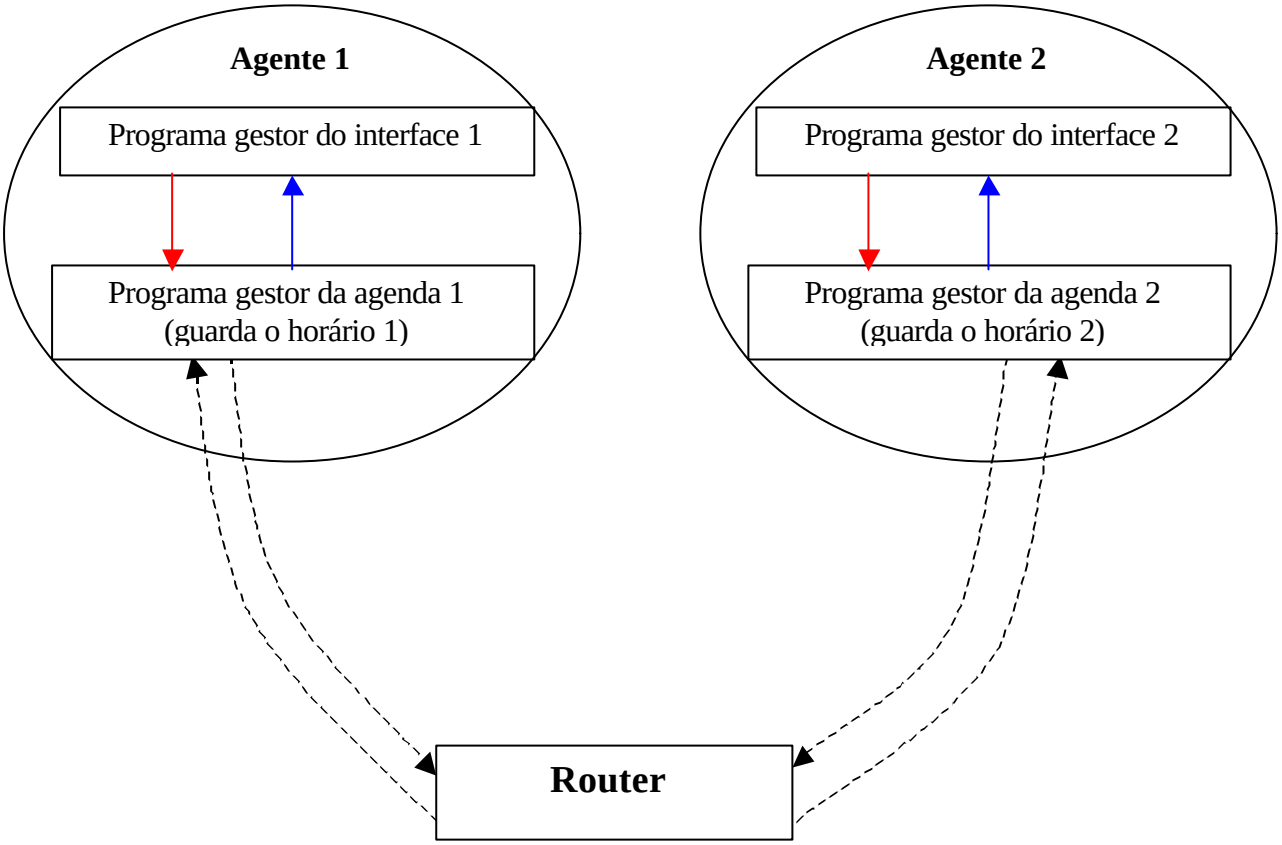
- Faz a interacção com o utilizador através de uma interface gráfica.
- Faz pedidos ao processo gestor da agenda correspondente e recebe as respectivas respostas.

Processo gestor da agenda:

- Inicialmente regista-se no router indicando a sua localização.
- No fim informa o router que se irá desligar.
- Comunica com o processo responsável pelo interface respondendo a este directamente.
- Comunica com os outros processos que gerem agendas enviando as respostas através do router.
- Guarda e gere o horário.

Router:

- Responsável por gerir as localizações dos processos gestores das agendas.
- Satisfaz pedidos dos processos nele registados sendo o elo de ligação entre eles.



Vantagens:

- O router sabe facilmente a localização de todos os processos que têm acesso ao horário.
- O processo que gere o interface só tem de saber onde se encontra o respectivo processo gestor da agenda onde se pretende ligar.
- O processo da interface, destinado para a interacção com o utilizador, não necessita de estar ligado no momento das marcações de reuniões.

Desvantagens:

- Aumento da complexidade estrutural da aplicação.
- Aumento do número de comunicações devido à existência de comunicações dentro do próprio agente e de comunicações entre processos gestores de horários e router.

Inicialmente optamos por ligar os processos gestores de horários directamente uns aos outros, no entanto, a gestão das localizações destes torna-se muito complexa.

Por esta razão, resolvemos implementar o *esquema de ligação com a ajuda de um router*. Com este esquema o processo gestor da agenda pode estar a correr num local diferente do processo que gere a interface estando sempre acessível para marcação de reuniões. Por outro lado, se desenvolvessemos um processo para cada agente não teríamos disponível a funcionalidade de poder marcar reuniões com a interface desligada.

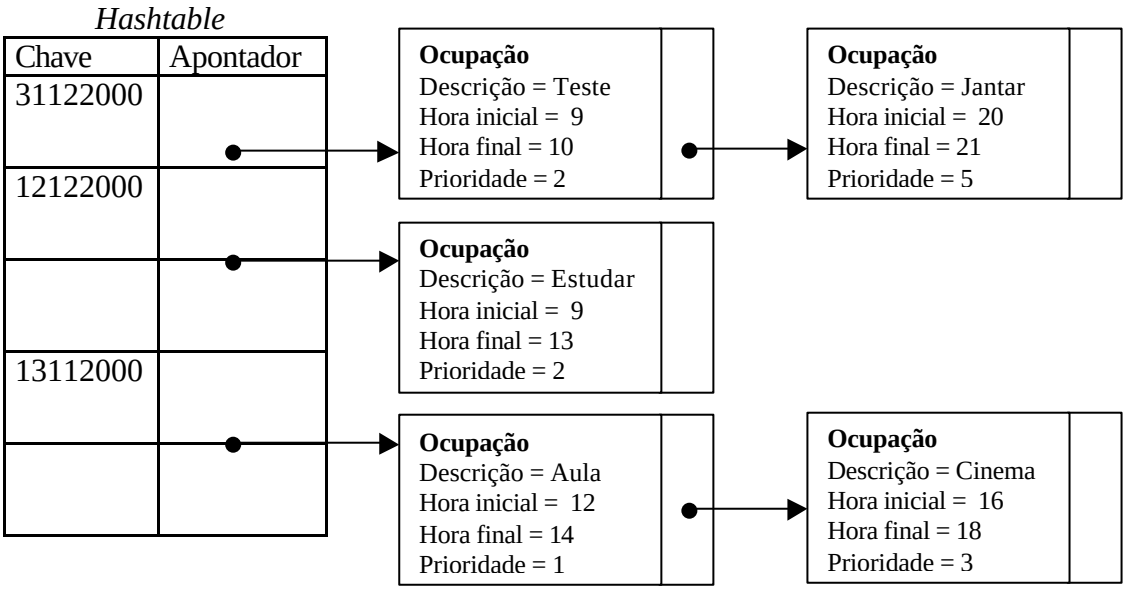
Para as mensagens processadas nas comunicações utilizou-se a linguagem KQML. O tipo de mensagens enviadas será explicado com mais detalhe na implementação.

Por último, a interface gráfica é composta por um horário que poderá ser actualizado semanalmente com a ajuda do rato. É pelo meio desta que se vão dar as ordens de marcação de reuniões. Pretende-se que a interface seja de utilização simples e rápida. Depois de passar a segurança é mostrado o horário correspondente à semana corrente.

É ainda de salientar que o mesmo processo gestor de interface tem a possibilidade de ligar a processos gestores de agendas diferentes. Com a funcionalidade de estabelecer uma nova ligação o horário actual é fechado e são pedidos novos dados: *login*, *password*, localização do processo gestor da agenda e porto correspondente.

Esquemas de representação de conhecimento

Para guardarmos a informação relativa a um horário no processo que gere a agenda utilizamos o conceito de ocupações. Estas têm como informação a descrição, a hora inicial, a hora final e a sua prioridade. Cada ocupação será um elemento de uma lista, lista esta que representa um dia. As ocupações são inseridas na lista ordenadas pelas horas. As listas são colocadas numa *hashtable* cuja chave é a data do dia em causa como se pode ver na figura.



Esta *hashtable*, sempre que o processo de interface se desliga, é guardada num ficheiro que é lido pelo processo gestor de horários sempre que é inicializado.

O processo gestor do interface contém uma base de conhecimento idêntica à do gestor da agenda. No entanto, não possui o ficheiro para guardar os dados pois estariam acessíveis a toda a gente. Sendo assim, o processo gestor do interface começa sempre com uma *hashtable* vazia. Como só é possível ao utilizador ver o horário semanalmente, o processo responsável pelo interface vai pedindo ao processo responsável pelo horário as ocupações em cada semana à medida das suas necessidades.

Este último, só pede as ocupações de uma semana se esta ainda não tiver sido requerida. Assim, cria-se uma espécie de memória *cache*. Esta é sempre tida em conta de modo a não existirem inconsistências entre a representação dos horários nos dois processos.

A escolha da estrutura adoptada deveu-se em grande parte ao facto do utilizador poder ter dias em que não existem actividades marcadas. Logo, se por exemplo estivéssemos a usar uma matriz (com as horas nas linhas e os dias nas colunas), esta iria desperdiçar grandes quantidades de memória. Por outro lado, as características da *hashtable* minimizam esse desperdício. Para que o número de entradas na tabela de dispersão não fosse extremamente elevado (originando mais colisões) decidiu-se organizar cada entrada por dia. Esta opção torna a determinação da chave muito simples que é a utilização da data do dia.

Outra hipótese seria usar apenas listas para a representação dos horários, mas estas tornariam-se extremamente extensas o que levaria a uma grande ineficiência na sua pesquisa.

Quanto ao conhecimento nas comunicações utilizam-se dois ficheiros, um para cada processo, que irão gerir a informação relativa a esse domínio.

Resumidamente, as aplicações criadas precisarão da seguinte informação que se encontra nos locais indicados:

Processo gestor do interface

Nome ⇒ login introduzido
Localização ⇒ obtida pelo processo com ajuda da classe InetAddress
Porto ⇒ indicado no ficheiro

Processo gestor da agenda

Login e *password* do processo interface correspondente ⇒ indicado no ficheiro
Porto para comunicar com o processo gestor da interface ⇒ indicado no ficheiro
Porto do processo gestor da agenda ⇒ indicado no ficheiro
Localização do processo gestor da agenda ⇒ obtida através da classe InetAddress
Endereço do "Router" e "RouterRegistar" ⇒ indicado no ficheiro

É ainda necessário controlar um conjunto de ficheiros para o correcto funcionamento do router do JATLite. Estes são inicializados na instalação do JATLite podendo ser posteriormente modificados.

Quanto ao processo que gere o horário, utilizará comunicações diferentes com portos diferentes para comunicar com o processo responsável pelo interface e para comunicar com o router.

Implementação

A descrição da implementação será dividida nos quatro módulos principais atrás explicados.

Manipulação da agenda

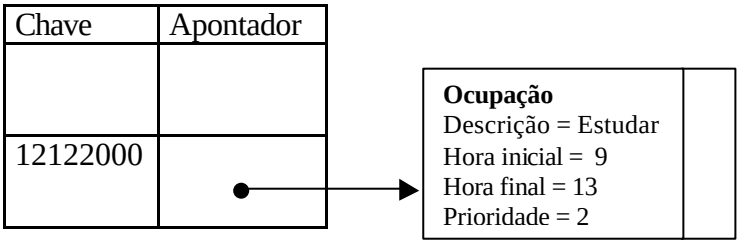
As operações possíveis sobre os horários são as de inserção, remoção e alteração de ocupações. Existe ainda a operação de gravar, contudo esta é feita sobre a tabela de dispersão que contém listas constituídas por ocupações.

Na introdução de ocupações começamos por verificar se o elemento da *hashtable* indicado pela chave (data da ocupação) já possui alguma ocupação. Se não possuir nenhuma é apenas necessário criar uma lista com a nova ocupação e introduzi-la na tabela de dispersão. Caso contrário temos que pesquisar na lista de ocupações existente o local de introdução da nova ocupação visto que estas encontram-se ordenadas pelas horas.

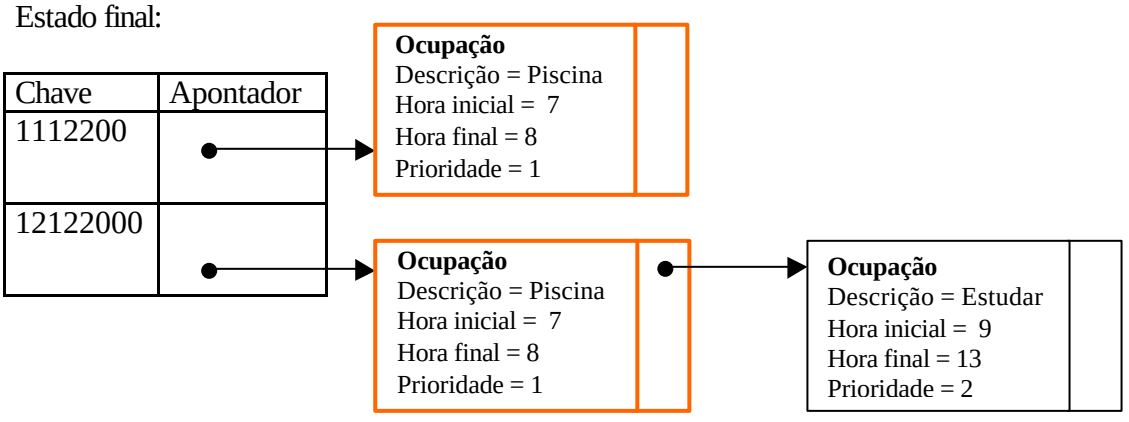
Exemplo de inserir ocupação ("ir à piscina das 7 às 8") no dia 11/12/2000 que ainda não tem nenhuma ocupação e outra no dia 12/12/2000 que já tem ocupações.

Estado incial:

Hashtable



Estado final:



Quanto à remoção se existir apenas um elemento na lista apaga-se a lista da *hashtable*. Se por outro lado, existirem vários elementos na lista apenas se retira a ocupação pretendida.

Assim, se apagasse-mos as ocupações a vermelho do estado final inseridas no exemplo anterior voltávamos ao estado inicial.

Esquema de negociação

O esquema de negociação desenvolvido baseia-se na média das indisponbilidades dos utilizadores. Começa-se por pedir ao utilizador que quer marcar a reunião o dia, a duração, a hora pretendida. O algoritmo irá pedir a todos os intervenientes a prioridade das suas ocupações a essa hora que vão de 1 a 5. Caso estes não tenham ocupações será indicada uma prioridade de 0. Se houver outras horas possíveis, as respostas são acompanhadas por uma hora alternativa e respectiva prioridade. As horas alternativas são aquelas que ainda não foram enviadas e cujo valor de indisponibilidade é mínimo. Ou seja, as agendas que respondem durante uma marcação guardam um historial das horas que já foram enviadas para a agenda marcadora. Para isso, no inicio da negociação é criado um array com as prioridades de cada hora do dia a marcar. Quando a prioridade de uma hora é enviada o seu valor no array passa a –1. Assim torna-se fácil de saber, quando se tem que escolher uma hora alternativa, qual a que tem menor prioridade.

Array inicial com as prioridades da agenda num determinado dia:

Horas	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Prior.	1	4	3	0	1	5	3	1	2	5	5	5	0	0

É pedida a hora 10 pela agenda marcadora. A resposta é enviada com uma alternativa, que neste caso serão as 19 horas. Então o historial fica alterado da seguinte forma:

Horas	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Prior.	1	4	3	-1	1	5	3	1	2	5	5	5	-1	0

Para saber qual é a hora com menor prioridade acha-se o mínimo do array sem contar com as horas cuja prioridade é –1.

Consideremos agora, a título de exemplo, que a agenda marcadora possui o seguinte horário:

Horas	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Prior.	1	5	0	0	2	2	4	1	1	1	0	0	5	4

Para auxiliar nos cálculos a agenda que marca a reunião, guarda uma média que no início corresponde às prioridades desta a dividir pelo número de pessoas intervenientes. Além disso, é feito um historial das respostas recebidas para nunca pedir a mesma hora aos outros processos. O historial é gerido através de uma matriz inicializada da seguinte forma, se tivermos três elementos intervenientes: uma agenda marcadora mais duas agendas:

Horas	7	8	9	10	11	12	13	14	15	16	17	18	19	20
César	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Gena	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Média	0,33	1,66	0	0	0,66	0,66	1,33	0,33	0,33	0,33	0	0	1,66	1,33

Consideremos no exemplo que a média máxima de indisponibilidades é de 1.

Quando chega uma resposta esta é colocada no seu devido local e a nova média é actualizada. No fim deste cálculo verifica-se se toda a coluna correspondente a essa hora já foi preenchida. Em caso afirmativo testa-se se a média obtida é inferior à indicada pelo utilizador. Se este teste for verdadeiro o algoritmo acaba. Se não é requerida, à agenda que enviou a resposta, a prioridade da hora cuja a média é a mais baixa e ainda não tenha sido pedia. Para saber se uma hora já foi ou não pedida a uma determinada agenda basta verificar se o valor correspondente na matriz é –1.

Após algumas iterações do algoritmo chegou-se à seguinte matriz:

Horas	7	8	9	10	11	12	13	14	15	16	17	18	19	20
César	0	-1	0	-1	5	2	-1	-1	-1	3	-1	-1	-1	-1
Gena	5	1	-1	-1	-1	3	-1	4	4	4	-1	-1	-1	-1
Média	2	2	0	0	2,33	2,33	1,33	1,66	1,66	2,66	0	0	1,66	1,33

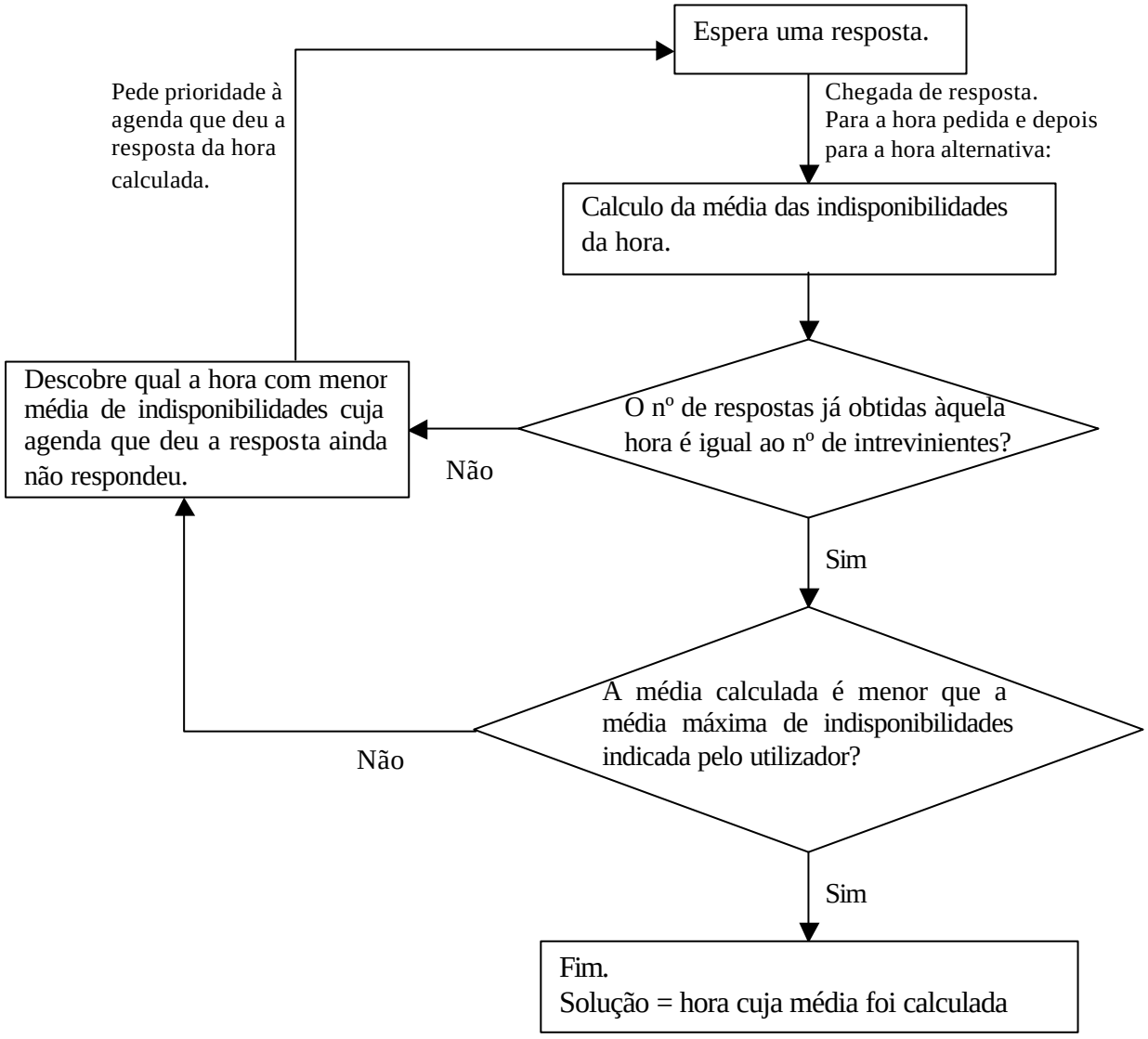
Neste caso, as médias das horas 7, 12 e 16 já foram testadas e não são inferiores à média máxima de indisponibilidades, logo, a marcação ainda não terminou. Se o último valor a ser acrescentado fosse o das 8 horas da Gena, seria pedida, a esta, a hora 9 pois é uma das que tem menor média e ainda não foi pedida. Então a matriz, após a chegada dessa resposta, seria a seguinte:

Horas	7	8	9	10	11	12	13	14	15	16	17	18	19	20
César	0	-1	0	-1	5	2	-1	-1	-1	3	-1	-1	-1	-1
Gena	5	1	1	-1	-1	3	-1	4	4	4	-1	-1	-1	-1
Média	2	2	0,33	0	2,33	2,33	1,33	1,66	1,66	2,66	0	0	1,66	1,33

Ao chegarmos a este ponto o processo terminaria pois a coluna das 9 horas encontra-se toda preenchida e o valor da média é inferior ao especificado. Seria enviada para o ecrã a resposta devida.

Fluxograma

Inicialmente pergunta-se a todas as agendas quais as indisponibilidades à hora pretendia pela agenda marcadora. Por cada resposta:

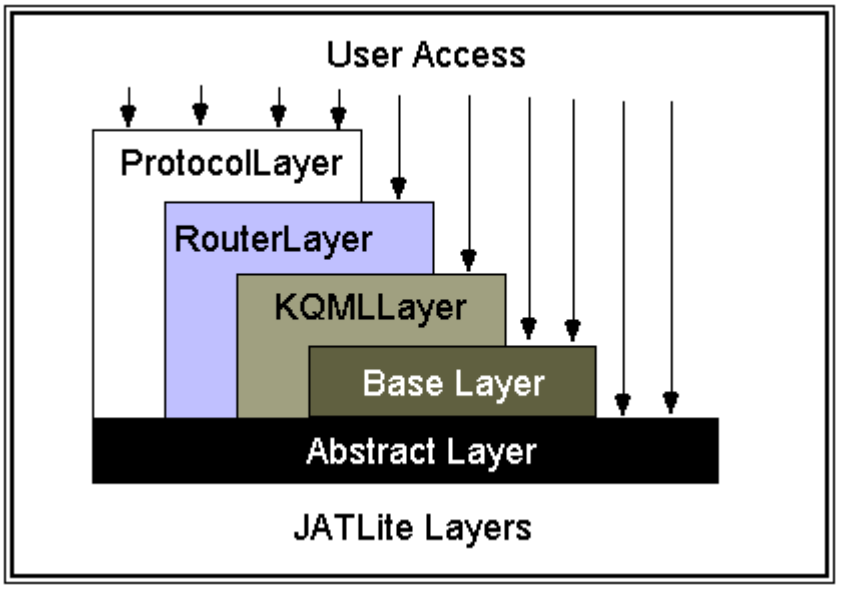

Espera uma resposta.
Chegada de resposta.
Para a hora pedida e depois para a hora alternativa:
Calculo da média das indisponibilidades da hora.
O nº de respostas já obtidas àquela hora é igual ao nº de intrevinientes?
Não
Descobre qual a hora com menor média de indisponibilidades cuja agenda que deu a resposta ainda não respondeu.
Pede prioridade à agenda que deu a resposta da hora calculada.
Sim
A média calculada é menor que a média máxima de indisponibilidades indicada pelo utilizador?
Não
Sim
Fim.
Solução = hora cuja média foi calculada

Comunicações

Como atrás descrito, para implementar a interacção entre os processos usou-se o JATLite. Usaram-se dois tipos de comunicação a intra-agente, entre o processo gestor do interface e o gestor da agenda, e as inter-agente, entre os processos gestores da agenda através do router.

A primeira realizou-se através da extensão do KQMLActionAgent enquanto a segunda do RouterClientAction. Como se pode verificar pela figura a interacção intra-agente tem menos funcionalidades pois é de uma camada inferior à da inter-agente. Esta

escolha foi propositada pois os diálogos internos ao agente não necessitam de um intermediário (router), tornado a sua execução menos pesada.



Na aplicação que gere a agenda existe uma classe que trata cada tipo de comunicação. Logo, este necessita de pelo menos um porto para transmitir e receber mensagens do processo que trata do interface e de outro para o fazer com o router.

Na criação das comunicações intra-agente à que seguir um conjunto de passos. Para começar temos que inicializar um conjunto de variáveis tais como a tabela de endereços, a das conecções e o buffer das mensagens. Além disso, deve-se inicializar a classe *security* que nos irá testar o protocolo de ligação e actualizar as variáveis atrás descritas sempre que necessário. De seguida, há que ler os ficheiros que contêm os dados acerca das ligações e com a ajuda da classe *InetAddress* reconhecer qual o *host* em que nos encontramos. Esta informação tem que ser introduzida na tabela de endereços para que se possa estabelecer uma ligação. O procedimento descrito é realizado tanto no processo gestor da interface como da agenda. Como é o gestor da agenda que tem que estar à espera que o do interface se ligue, este tem que abrir um *thread* de escuta. O denominado *ServerThread* quando recebe uma mensagem cria de imediato um *ReceiverThread* para a tratar, ficando à espera de novas mensagens.

Focando agora a nossa atenção no diálogo inter-agente, também temos que seguir uma sequência de operações. Primeiro insere-se os seguintes endereços na respectiva tabela: o do próprio gestor do horário, o do router e do router registrar (responsável por registar as agendas). Depois, é também preciso criar um *ServerThread* para escutar a rede, registar os processos no router e ligar ao router com a ajuda do procedimento *connect*.

Para enviar as mensagens KQML não existem diferenças nestes dois tipos de interacção. Então criou-se, em cada classe, um procedimento genérico que recebe como argumentos o tipo de mensagem (*ask-if*, *reply*, *tell*, etc), o emissor, o receptor e o conteúdo da mensagem. O retorno desta função é verdadeiro ou falso conforme o sucesso do envio da mensagem.

```
public boolean msgKQML(String tipo,String emissor,String receptor, String msg)
{
  //criação de uma mensagem KQML
  KQMLmessage mKQML = new KQMLmessage();
  mKQML.addFieldValuePair("performative",tipo);
  mKQML.addFieldValuePair("sender",emissor);
  mKQML.addFieldValuePair("receiver",receptor);
  mKQML.addFieldValuePair("content",msg);

  //envio da mensagem
  try
  {
    sendMessage(mKQML);
    return true;
  }
  catch (ConnectionException e)
  {
    System.out.println(e.toString());
  }
  return false;
}
```

Quanto à recepção das mensagens, quando uma mensagem nova chega o procedimento *act* é imediatamente executado. Este contém um conjunto de clausulas *if* que interpretam o pedido ou a informação da mensagem encaminhando-a para um procedimento. Cada procedimento, escolhido para tratar a informação, irá satisfazer um pedido, guardar informação ou ainda utiliza-la em algoritmos.

É de salientar que as mensagens KQML via inter-agentes são encapsuladas sobre um KQMLmail, logo, para extrair os dados da mensagem procede-se de uma forma ligeiramente diferente. A diferença consiste no "desembrulhar" do KQMLmail transformando-o numa mensagem KQML. Além disso, é necessário pedir ao router para apagar a mensagem no seu buffer através do procedimento (`addToDeleteBuffer`).

As mensagens usadas para pedir serviços ou retribuir respostas são as seguintes:

Interface-Agenda

Tipo	Mensagem	Função
perform	newConnection(nome, *host*, porto, tipo)	Cria um novo endereço na respectiva tabela do processo gestor da agenda.
perform	confirmMeeting(descrição, data, duração)	Confirma a marcação da solução obtida com os outros intervenientes.
perform	CancelMeeting	Cancela a busca de solução.
perform	Other	Procura outra solução.
ask	week(número relativo da semana)	Pede ao processo gestor da agenda as ocupações da semana correspondente. Como o número é relativo se este for 0 é a semana actual (< 0 semanas anteriores à actual e > 0 semanas posteriores à actual).
ask	ConnectedUsers	Pede ao processo gestor da

		agenda o nome das agendas dos utilizadores actualmente ligados.
ask	newMeeting(data, duração, utilizadores)	Pede ao processo gestor da agenda para marcar uma nova reunião com os utilizadores especificados.
ask-if	login(login, password)	Perguntar ao processo gestor da agenda se o login e password estão correctos.
tell	newOcupation(data, descrição, hora inicial, hora final, prioridade)	Indica ao processo gestor da agenda que foi criada uma nova ocupação.
tell	deleteOcupation(data, descrição, hora inicial, hora final, prioridade)	Indica ao processo gestor da agenda que foi apagada uma ocupação.
tell	Bye	Indicar ao processo gestor da agenda que o processo do interface vai-se desligar.

Agenda-Interface

Tipo	Mensagem	Função
perform	stdout1(emissor, msg)	Pede ao processo gestor da interface para indicar na caixa de texto de mensagens enviadas da janela de comunicações a mensagem indicada por msg.
perform	stdout2(emissor, msg)	Pede ao processo gestor da interface para indicar na caixa de texto de mensagens recebidas da janela de comunicações a mensagem indicada por msg.
perform	solution(hora, grau de indisponibilidade)	Pede ao processo gestor da interface para colocar a solução encontrada no ecrã.
reply	login(valor booleano)	Responde ao processo gestor da agenda se o login e password estão correctos.
reply	week(ocupações)	Responde ao pedido do processo gestor da agenda de saber quais as ocupações numa determinada semana.
reply	connectedUsers(utilizadores)	Responde ao pedido do processo gestor da agenda de saber quais os utilizadores ligados no momento.

Agenda-Router-Agenda

Tipo	Mensagem	Função
perform	ConfirmedMeeting(hora, descrição, data, duração)	Confirma a marcação da reunião na hora indicada.
tell	newMeetingDate(data, duração)	Indica ao outro processo gestor da agenda que irá começar uma negociação para marcar uma reunião no dia indicado.
ask	meeting(hora)	Pede ao outro processo gestor da agenda para lhe enviar a prioridade na hora indicada e respectiva alternativa (se existir).
reply	meetingAnswer(hora, prioridade, "no")	Resposta do processo gestor da agenda à hora com a prioridade da hora pedida, não existindo uma alternativa a dar ao outro processo gestor da agenda.
reply	meetingAnswer(hora, prioridade, "yes", hora alternativa, prioridade à hora alternativa)	Resposta do processo gestor da agenda à hora com a prioridade da hora pedida dando uma alternativa.
tell	Hello	Mensagem auxiliar enviada ao outro processo gestor da agenda através do router que apenas verifica se este ainda se encontra ligado. Não se pretende obter resposta.

Para o processamento do conteúdo das mensagens elaboraram-se duas funções. Uma que retorna o primeiro argumento da mensagem e outra que o retira. Assim, é possível aceder a todos os argumentos facilmente.

Interface

A implementação da interface teve vários pontos essenciais. Foram criadas várias janelas de interacção com o utilizador que tentam dar ao programa uma simples utilização.

Antes de criar a tabela que representa o horário semanal, teve-se que elaborar um modelo desta (*AbstractTableModel*) que irá inicializar a primeira coluna com as horas do dia e a primeira linha com os dias da semana. Sobre a tabela é possível fazer selecções de modo a facilitar a inserção, alteração e remoção de ocupações.

Foram desenvolvidos dois *array's* para as agendas ligadas que aparecem na janela de marcação de reuniões. Estes são os responsáveis pela visualização correcta dos utilizadores seleccionados para a marcação de uma reunião.

Uma funcionalidade muito interessante é a criação de um sistema tipo *cache* para a visualização das ocupações nas diversas semanas. Este sistema é implementado graças a duas variáveis. Estas indicam o intervalo de semanas que já foram pedidas ao processo gestor da agenda visto que o horário localiza-se neste. Se a semana actual se encontrar entre o intervalo definido por estes dois valores, as ocupações desta semana não necessitam de ser carregadas do processo gestor da agenda, pois já estão na *cache* do

processo do interface. Para não haver inconsistências todas as alterações realizadas sobre ocupações neste último são prontamente notificadas ao processo gestor da agenda.

No arranque do interface o horário encontra-se vazio existindo um pedido à aplicação que gere a agenda para enviar as ocupações da semana actual. Quando a interface é desligada a *cache* é esquecida não havendo possibilidade de terceiros a visualizarem.

Todos os campos introduzidos na interacção através da interface são testados pois podem conter valores inválidos. Caso isso aconteça é sempre enviado para o ecrã uma janela de erro.

Análise da complexidade do algoritmo

Achamos que se trata de um algoritmo suficiente, que tenta reduzir o número de mensagens enviadas pelo processo marcador. Chegamos à conclusão que por dia, por cada processo gestor da agenda, na pior das hipóteses são enviadas: (horas do dia/2) mensagens.

Mesmo com muitos agentes, o cálculo das médias nunca será muito trabalhoso visto que o número deste também nunca deverá ultrapassar as centenas.

Como estamos a receber mensagens pelo meio da rede estas vão chegando mais ou menos aleatoriamente e como temos *threads* a tratar de cada uma podemos dizer que o nosso algoritmo é de certo modo multiprogramado. Por isso, também houve a necessidade de declarar alguns procedimentos como *synchronized* para que não houvesse problemas relacionados com a multiprogramação.

Um algoritmo que apenas calculasse a solução óptima não teria graça pois o número de mensagens seria sempre o mesmo, ou seja, o processo gestor do horário teria sempre que saber quais as indisponibilidades a todas as horas de todas as outras agendas.

Recursos

Bibliografia

- "Artificial Intelligence: a modern aproach". Stuart Russel e Peter Norvig, 1995, Prentice Hall.
- Acetatos fornecidos pelos docentes da cadeira de Sistemas Periciais e Robótica.
- Documentação fornecida conjuntamente com as ferramentas utilizadas no desenvolvimento desta aplicação, www.java.stanford.edu.
- Api e tutorial do Java.

Ferramentas de desenvolvimento

- JDK 1.2
- JATLite
- Windows 98/NT

Especificação do sucesso

Um programa deste tipo deverá satisfazer os seguinte itens:

- **Interface amigável** – O interface deverá ser o mais intuitivo possível, de modo a que o utilizador se sinta o mais à vontade a trabalhar no sistema.
- **Robustez** – O programa deverá ser o mais robusto possível, isto é, deverá detectar o maior número de situações de erro, prevenindo-os ou recuperando-os.
- **Rapidez** – O sistema terá de produzir uma resposta dentro de um tempo aceitável.
- **Qualidade** – Tão importante como a rapidez de resposta, é que essa mesma resposta satisfaça a totalidade dos agentes nela presentes, ou na impossibilidade desta situação que satisfaça o maior número de utilizadores possível.
- **Portabilidade** – Será do interesse para futuros utilizadores que possam recorrer à aplicação a partir de vários locais, acedidos por variadas plataformas.

Como existe um trabalho que foi realizado sem a utilização do JATLite seria interessante compará-los. Alguns dos testes a realizar seriam comparar o tempo de resposta, o número de mensagens médio por resposta, o grau de satisfação das respostas e os interfaces.

Além disso, também seria de verificar as grandes vantagens de poder ter um processo gestor do horário, tal como o que foi implementado, a correr em *background* num sistema permanentemente ligado à internet.

Conclusão

Concluímos que esta arquitectura de negociações e cooperações entre agentes tem muita aplicabilidade prática principalmente agora que a internet está cada vez mais acessível. Como exemplo de um sistema deste tipo temos os leilões virtuais.

Por outro lado, a comunicação entre agentes pode ser aplicada noutras áreas como a robótica que leva a soluções muito interessantes.

Ao optarmos pelo *esquema de ligação com a ajuda de um router* deparamo-nos com um esquema de comunicações bastante complexo, o que nos trouxe alguns problemas que foram prontamente estudados e resolvidos.

O recurso a novas ferramentas de desenvolvimento não se revelou de modo algum um obstáculo. A leitura prévia da documentação existente e alguns esclarecimentos por parte dos docentes da cadeira foram suficientes para um bom conhecimento das ferramentas. A incorporação de módulos de comunicação com o JATLite abriu-nos um vasto leque de possibilidades para a elaboração do trabalho.

Com o evoluir do trabalho foram surgindo pequenos contratempos, cuja sua resolução foi gratificante para o grupo, na medida em que se aprenderam e aplicaram novas soluções.

Para finalizar é de referir que o trabalho foi sempre seguido com alguma expectativa, pois à medida que o íamos realizando, íamos tendo uma noção da interacção criada entre os agentes. A visualização desta foi fundamental para a percepção de certos problemas criados num sistema deste género.

Melhoramentos

Como melhoramentos temos a sugerir implementar outro tipo de algoritmos: que melhorem a estratégia de negociação, que sejam mais eficazes e que possuam algumas condições de paragem diferentes, como o número de mensagens enviadas ou o tempo de espera máximo. Estes dois parâmetros referidos até seriam de fácil implementação no nosso algoritmo, porém o tempo este ano foi mais limitado o que não nos proporcionou esse desenvolvimento.

Também seria interessante colocar um intervalo de datas de pesquisa da solução de modo a que o utilizador pudesse escolher entre que datas queria marcar a reunião.

Seria de corrigir certos "bugs", como por exemplo, a da selecção na tabela do horário através do rato que deixa seleccionar linhas e colunas, em vez de deixar seleccionar apenas uma coluna de cada vez.

Existe ainda um pequeno pormenor que seria de corrigir que era não deixar efectuar marcações em simultâneo. Ou seja, quando uma marcação estivesse em curso não deixar que outro utilizador pudesse fazer outra se um ou mais dos intervenientes entrasse em ambas.

Manual do Utilizador

Manual de Instalação

Para a correr o programa não é necessário instalar previamente o JATLite visto que as classes usadas deste já estão incluídas no programa.

Antes de iniciar a execução do programa pode ser necessário configurar os ficheiros *AgenteC* para os gestores do interface e *AgenteS* para os gestores da agenda. Cada gestor da agenda convém ter pelo menos um nome diferente.

Em principio só será necessário configurar a máquina onde corre o router, o login e password. Para isso altera-se os campos a "bold" do ficheiro *AgenteS* do gestor da agenda:

AgenteS

```
# nome do gestor da agenda
Antonio
#password
spr
# porto da agenda para o interface - AgenteS
6666
# porto da agenda para o router - AgenteS
6667
# password
(agent-info :password a)
# dados do router - AgenteR
Router,coyote-ugly,6663,RouterServerThread,(agent-info :agent-name AMR)
RouterRegistrar,coyote-ugly,6664,RegistrarServerThread,(Registrar)
```

Todos os processos atrás referidos, encontram-se ou em directórios diferentes do mesmo computador ou em computadores diferentes. Caso estejam no mesmo, os seus portos não podem ser iguais para que se possa facilmente distinguir os destinatários das mensagens. Para uma modificação dos portos pode-se alterar o ficheiro anterior, o do *AgenteC* do interface e do router. De seguida são mostrados esses ficheiros com a "bold" os portos que podem ser alterados:

AgenteS

```
# nome do gestor da agenda
Antonio
#password
spr
# porto da agenda para o interface - AgenteS
6666
# porto da agenda para o router - AgenteS
6667
# password
(agent-info :password a)
# dados do router - AgenteR
Router,coyote-ugly,6663,RouterServerThread,(agent-info :agent-name AMR)
RouterRegistrar,coyote-ugly,6664,RegistrarServerThread,(Registrar)
```

AgenteC

```
# porto do gestor do interface - AgenteC
6665
```

Para alterar os portos do router é preciso configurar o ficheiro *AddressFile* que se encontram no directório */RouterLayer/Resource* do router.

AddressFile

```
# This is an example file for the static Agent Addresses
# AgentID, Host InetAddress, Port no, Type, Description
# Since this is ServerThread, you do not need to specify InetAddress
Router,null,6663,RouterServerThread,(agent-info :agent-name AMR)
RouterRegistrar,null,6664,RegistrarServerThread,(Registrar)
```

Não esquecer ainda de colocar na *classpath* o directório corrente através do comando:

```
set CLASSPATH=%CLASSPATH%;.
```

Após isto é preciso pôr a correr o router através do seguinte comando executado no directório deste:

```
java RouterLayer.Router.RouterAction RouterLayer/Resource/routerscript
```

De seguida é necessário executar os processos gestores dos horários com o seguinte comando para cada um deles:

```
java AgendaS
```

Para podermos começar a interagir com o programa é preciso criar pelo menos um interface:

```
java AgendaC
```

Manual de Utilização

Agora que inicializamos a nossa aplicação temos que introduzir um login e uma password especificados anteriormente no ficheiro *AgenteS* do processo gestor da agenda correspondente. Além disso, temos que indicar a localização do *host* deste último, pelo meio do nome do computador e respectivo porto.

Ao entrar no programa propriamente dito se quisermos adicionar uma nova ocupação, basta seleccionar na tabela as horas e depois carregar sobre o botão cujo icon é um "pato". Existe ainda outra maneira de realizar esta operação que é carregar logo no icon sem seleccionar nada, contudo é preciso introduzir mais informação pelo teclado.

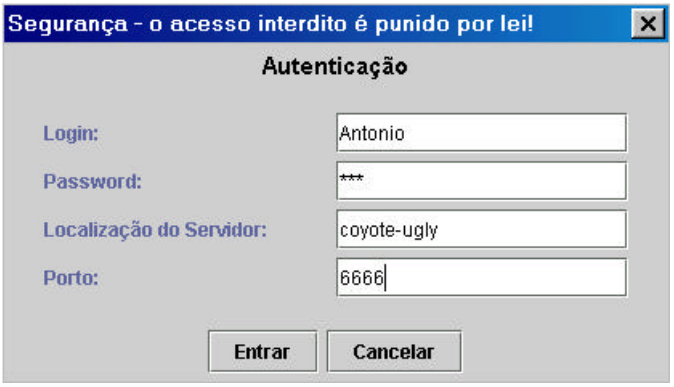
Para remover uma ocupação basta selecciona-la e carregar no botão de apagar indicado por uma cruz.

Quanto à marcação de reuniões pode ser feita através do botão com o icon do mundo. Depois, é necessário introduzir os dados relativos a esta indicando o grau médio máximo de indisponibilidade pretendida. Os utilizadores com quem queremos marcar as reuniões podem ser seleccionados com a ajuda das setas que se encontram no ecrã. Finalmente, ao carregarmos no botão marcar aparece uma nova janela que mostra as comunicações efectuadas entre as agendas. Ao ser encontrada uma solução, é possível pedir outra ou confirma-la. Neste último caso, as reuniões serão automaticamente marcadas nas agendas dos intervenientes.

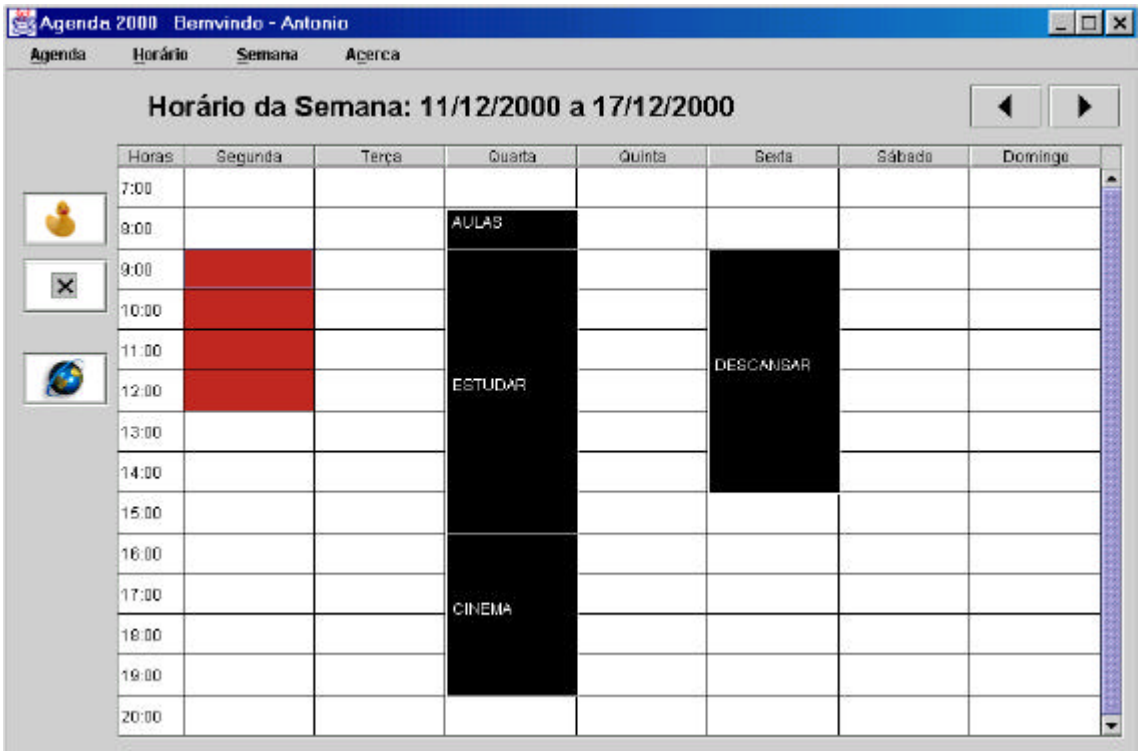
Se quisermos visualizar os nomes dos autores do programa podemos faze-lo através do menu carregando sobre as palavras "acerca de".

Exemplo de uma Execução

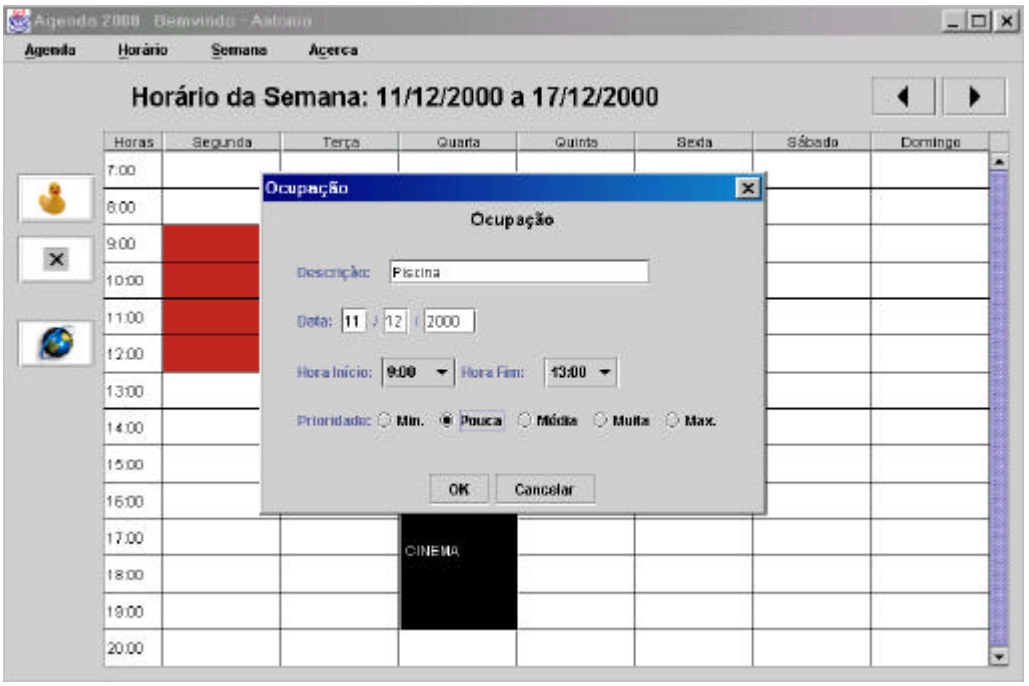
Janela inicial de segurança e identificação do utilizador. Como já foi dito, nesta introduz-se o login, password, localização do processo que gere a agenda e seu porto.



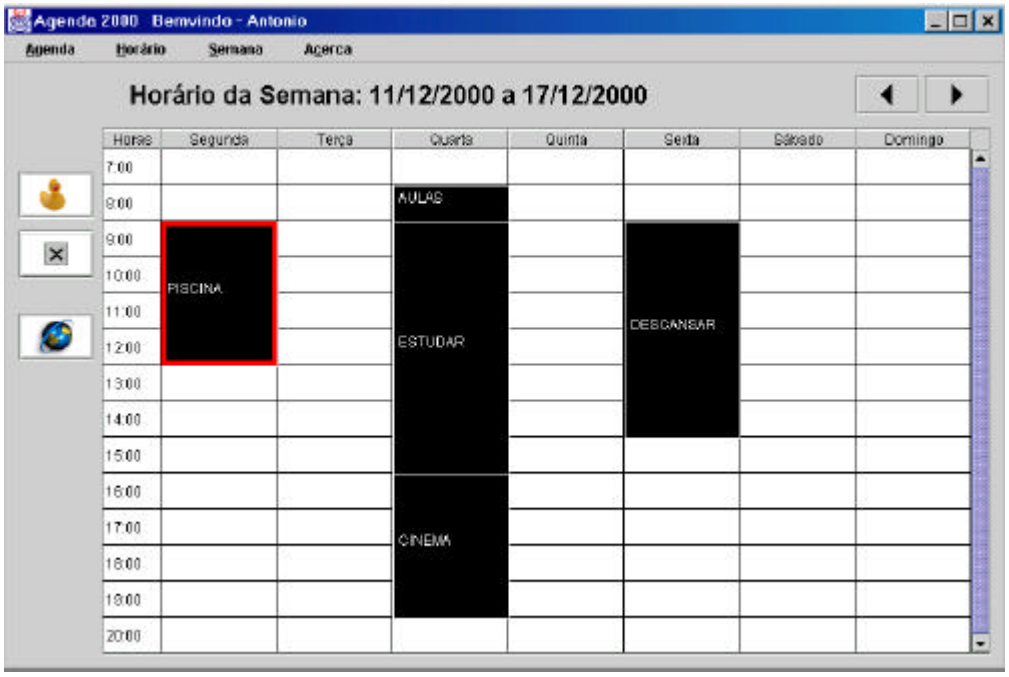
A imagem seguinte mostra a janela principal do programa havendo uma selecção efectuada pelo rato na segunda-feira das 9 às 13.



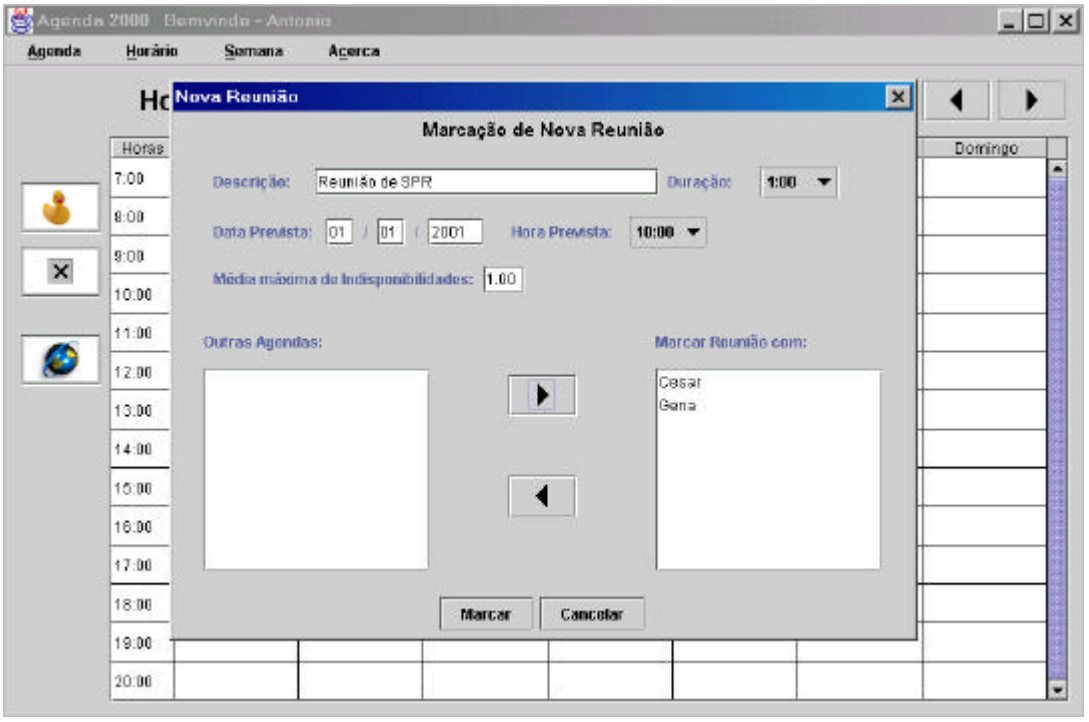
Se no momento da figura anterior se carregar no botão cujo o icon é um “pato” quer dizer que queremos introduzir uma nova ocupação. Os campos de hora início e hora fin já se encontram preenchidos bastando digitar a descrição e indicar a prioridade dessa ocupação.



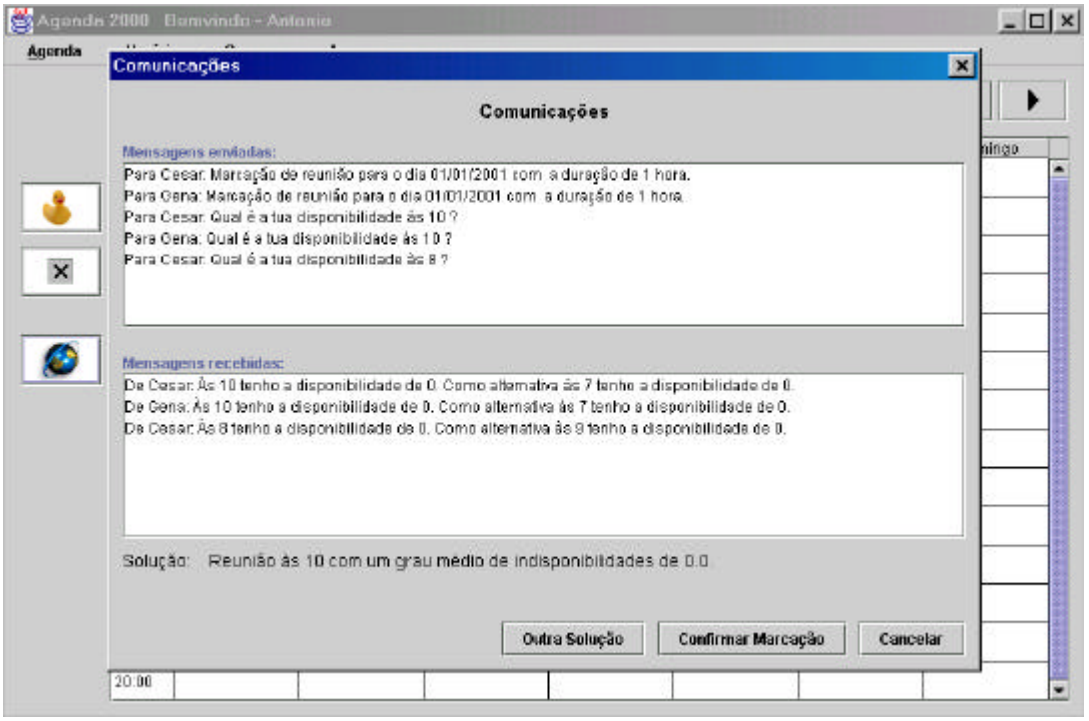
Ao carregar sobre o botão “OK” a nova ocupação aparece. Se pressionarmos sobre esta seleccionamo-la (aparecendo um risco vermelho à volta) e ao invocarmos de novo o “pato” podemos alterar as suas definições. Por outro lado, se a quisermos apagar basta carregar sobre o botão que contém um cruz que a selecção é removida do horário.



Para criar uma reunião pressiona-se sobre o botão cujo icon é um mundo, aparecendo a seguinte janela que precisa de ser preenchida:



Ao carregar sobre o botão "Marcar" uma nova janela é aberta mostrado as comunicações e a solução obtida para a marcação anterior quando é encontrada:


Agenda 2000 Bemvindo - Antonio
Agenda
Comunicações
Comunicações
Mensagens enviadas:
Para Cesar: Marcação de reunião para o dia 01/01/2001 com a duração de 1 hora.
Para Gena: Marcação de reunião para o dia 01/01/2001 com a duração de 1 hora.
Para Cesar: Qual é a tua disponibilidade às 10 ?
Para Gena: Qual é a tua disponibilidade às 10 ?
Para Cesar: Qual é a tua disponibilidade às 8 ?
Mensagens recebidas:
De Cesar: Às 10 tenho a disponibilidade de 0. Como alternativa às 7 tenho a disponibilidade de 0.
De Gena: Às 10 tenho a disponibilidade de 0. Como alternativa às 7 tenho a disponibilidade de 0.
De Cesar: Às 8 tenho a disponibilidade de 0. Como alternativa às 9 tenho a disponibilidade de 0.
Solução: Reunião às 10 com um grau médio de indisponibilidades de 0.0
Outra Solução
Confirmar Marcação
Cancelar
20:00

Aqui podemos pedir uma nova solução, confirmarmos a solução obtida ou ainda cancelarmos a marcação da reunião.

Listagem do Código

Ficheiros de configuração:

AgenteC

```
# porto do cliente - AgenteC
6665
```

AgenteS

```
# nome do servidor
Antonio
#password
spr
# porto do servidor para o cliente - AgenteS
6666
# porto do servidor para o router - AgenteS
6667
# password
(agent-info :password a)
# dados do router - AgenteR
Router,coyote-ugly,6663,RouterServerThread,(agent-info :agent-name AMR)
RouterRegistrar,coyote-ugly,6664,RegistrarServerThread,(Registrar)
```

RouterScript

```
# Router Name. The address file should have the Router address with the
Router name.
Router
# Address file(include Router, RotuerRegistrar, and other) path
D:\jatlite\RouterLayer/Resource/AddressFile
# Incoming mail box directory. should ended with '/'
D:\jatlite\RouterLayer/Resource/incoming/
# password file path.
D:\jatlite\RouterLayer/Resource/Password
# registry file path.
D:\jatlite\RouterLayer/Resource/Registry
# Maxiumum duration time for the Router Receiver Thread.
1000
# Maximum trial time for reconnecting to the stand alone off line agent
5
# Reconnection trial time period in minutes( here, 1 hr)
60
# Maximum trial time to send reserved message
5
# Sending reserved message time period
60
# Router agent action sleep time in milliseconds
5000
```

Código criado:

Cliente:

AgendaC.java

```
//A classe principal da aplicação cliente

class AgendaC
{
  public static JPass janelaPass;
  public static AgenteC cliente;

  public static void main(String[] argv)
  {
    cliente = new AgenteC();
    cliente.start();

    //janela para a password que depois abre a principal
    janelaPass = new JPass();
    janelaPass.setSize(420,240);
    //janelaPass.pack();
    janelaPass.setLocation(280,200);
    janelaPass.show();
  }
}
```

AgenteC.java

```
import BaseLayer.*;
import Abstract.*;
import KQMLLayer.*;

import java.io.*;
import java.net.*;

public class AgenteC extends KQMLAgentAction
{
  public static String idS;
  public static String localS;
  public static String portoS;
  public static String tipoS;

  public static String idC;
  public static String localC;
  public static String portoC;
  public static String tipoC;

  public static String utilizadores;

  public AgenteC()
  {
    super();

    String readline;

    try
    {
      // procedimentos de arranque
      _addresses = new BAddressTable();
      _connections = new BConnectionTable();
      _queue = new BMessageBuffer();
```

```
      // lê o ficheiro de endereco e coloca os seu dados numa tabela
      BufferedReader input = new BufferedReader(new FileReader(new
File("AgenteC")));

      //dados deste servidor
      while((readline = input.readLine()) != null)
      {
        if(!readline.startsWith("#"))
        {
          portoC = readline;
          break;
        }
      }

      input.close();
      // deve ser instanciado depois da tabela ter sido criada
      _security = new KQMLSecurity(_addresses);
    }
    catch (Exception e)
    {
      System.out.println(e.toString());
    }
  }

  public boolean ligaAoServidor(String local,String porto)
  {
    try
    {
      InetAddress localhost = InetAddress.getLocalHost();
      System.out.println("Localização actual: " + localhost.getHostName());

      idS = "Servidor";
      localS = local;
      portoS = porto;
      tipoS = "AgenteS";

      idC = "Cliente";
      localC = localhost.getHostName();
      tipoC = "AgenteC";
    }
    catch(Exception e)
    {
      System.out.println(e.toString());
      return false;
    }
    try
    {
      //como o id é igual nas várias tentativas o address é escrito sempre por
cima
      System.out.println("Servidor: " + idS + ',' + localS + ',' + portoS + ','
+ tipoS);
      Address addr = new Address(idS + ',' + localS + ',' + portoS + ',' +
tipoS);
      _addresses.addAddress(addr);
      idS=addr.getID();

      //como o id é igual nas várias tentativas o address é escrito sempre por
cima
      System.out.println("Cliente: " + idC + ',' + localC + ',' + portoC + ','
+ tipoC);
      addr = new Address(idC + ',' + localC + ',' + portoC + ',' + tipoC);
      _addresses.addAddress(addr);
      idC=addr.getID();
      setName(idC);
```

```
    }
    catch (Exception e)
    {
      System.out.println(e.toString());
      return false;
    }


if(!msgKQML("perform",idC,idS,"(newConnection("+idC+','+localC+','+portoC+','+
tipoC+"))"))
      return false;

    return true;
  }

  public void processMessage(String command,Object obj)
  {
    String[] args = (String[])obj;

    if(command.equals("UnsupportedType"))
    {
      System.out.println(command + " " + args[0]);
    }
    else
    {
      if(command.equals("InitializeFailed"))
      {
        System.out.println(command + " " + args[0]);
        endAction();
      }
    }
  }

  protected boolean Act(Object o)
  {
    System.out.println((String) o);

    try
    {
      KQMLmessage mKQML = new KQMLmessage((String) o);

      String tipo = mKQML.getValue("performative");
      String emissor = mKQML.getValue("sender");
      String conteudo = mKQML.getValue("content");

      if(tipo.equals("reply") && conteudo.startsWith("(week("))
        introduzirSemana(conteudo.substring(6));
      else if(tipo.equals("reply") && conteudo.startsWith("(connectedUsers("))
      {
        utilizadores = conteudo.substring(16);
        AlgC.resposta = 0;
      }
      else if(tipo.equals("perform") && conteudo.startsWith("(stdout1("))
        imprimeEmJComunic1(conteudo.substring(9));
      else if(tipo.equals("perform") && conteudo.startsWith("(stdout2("))
        imprimeEmJComunic2(conteudo.substring(9));
      else if(tipo.equals("reply") && conteudo.startsWith("(login("))
        trataRespLogin(conteudo.substring(7));
      else if(tipo.equals("perform") && conteudo.startsWith("(solution("))
        trataSolucao(conteudo.substring(10));

      return true;
    }
    catch(KQMLLayer.ParseException e)
```

```
  {
    System.out.println(e.toString());
    return false;
  }
}

public boolean msgKQML(String tipo,String emissor,String receptor,String msg)
{
  //criação de uma mensagem KQML
  KQMLmessage mKQML = new KQMLmessage();
  mKQML.addFieldValuePair("performative",tipo);
  mKQML.addFieldValuePair("sender",emissor);
  mKQML.addFieldValuePair("receiver",receptor);
  mKQML.addFieldValuePair("content",msg);

  //envio da mensagem
  try
  {
    sendMessage(mKQML);
    return true;
  }
  catch (ConnectionException e)
  {
    System.out.println(e.toString());
  }
  return false;
}

public boolean login(String login,String pass)
{
  if(msgKQML("ask-if",idC,idS,"(login(" + login + ',' + pass + "))"))
    return true;
  else return false;
}

public boolean logout()
{

  if(msgKQML("tell",idC,idS,"(bye)"))
  {
    (_connections.getConnection(idS)).endConn();
    return true;
  }
  else
  {
    _addresses.removeAddress(idS);
    return false;
  }
}

public void trataRespLogin(String conteudo)
{
  if(AlgC.proximoValor(conteudo).equals("true"))
    AlgC.resposta = 1;
  else
    AlgC.resposta = 0;
}

public boolean novaOcupacao(String conteudo)
{
  if(msgKQML("tell",idC,idS,"(newOcupation("+conteudo+"))"))
    return true;
  else return false;
}
```

```
  public boolean apagaOcupacao(String conteudo)
  {
    if(msgKQML("tell",idC,idS,"(deleteOcupation("+conteudo+"))"))
      return true;
    else return false;
  }

  public boolean pedeSemana(int s)
  {
    if(msgKQML("ask",idC,idS,"(week("+s+"))"))
      return true;
    else return false;
  }

  public void introduzirSemana(String conteudo)
  {
    int dia,mes,ano,horai,horaf,prior;
    String desc;

    while(conteudo.indexOf(",")!=-1)
    {
      dia = Integer.parseInt(AlgC.proximoValor(conteudo));
      conteudo = AlgC.retiraValor(conteudo);
      mes = Integer.parseInt(AlgC.proximoValor(conteudo));
      conteudo = AlgC.retiraValor(conteudo);
      ano = Integer.parseInt(AlgC.proximoValor(conteudo));
      conteudo = AlgC.retiraValor(conteudo);
      desc = AlgC.proximoValor(conteudo);
      conteudo = AlgC.retiraValor(conteudo);
      horai = Integer.parseInt(AlgC.proximoValor(conteudo));
      conteudo = AlgC.retiraValor(conteudo);
      horaf = Integer.parseInt(AlgC.proximoValor(conteudo));
      conteudo = AlgC.retiraValor(conteudo);
      prior = Integer.parseInt(AlgC.proximoValor(conteudo));
      conteudo = AlgC.retiraValor(conteudo);

      Ocup ocupacao = new Ocup(desc,horai,horaf,prior);
      AlgC.insereOcupacao(dia,mes,ano,ocupacao);
    }
    AlgC.resposta = 0;
    JAgenda.scrollpane.repaint();
  }

  public String pedeListaUtil()
  {
    msgKQML("ask",idC,idS,"(connectedUsers)");
    AlgC.esperaResposta();
    return utilizadores;
  }

  public void marcaReuniao(String reuniao)
  {
    msgKQML("ask",idC,idS,"(newMeeting(" + reuniao+ "))");
  }

  public void imprimeEmJComunic1(String conteudo)
  {
    String r = AlgC.proximoValor(conteudo);
    conteudo = AlgC.retiraValor(conteudo);
    String f = conteudo.substring(0,conteudo.indexOf('('));

    if(f.equals("meeting"))
    {
      AlgC.comunic.tEnviadas.append("Para " + r + ": Qual é a tua
disponibilidade às ");
```

```
      AlgC.comunic.tEnviadas.append(conteudo.substring(8,conteudo.length()-3) +
" ?\n");
    }
    else
    {
      conteudo = conteudo.substring(conteudo.indexOf('(')+1,conteudo.length()-
1);
      String dia = AlgC.proximoValor(conteudo);
      conteudo = AlgC.retiraValor(conteudo);
      String duracao = AlgC.proximoValor(conteudo);

      if(dia.length()==7)
        dia = '0' + dia.substring(0,1) + '/' + dia.substring(1,3) + '/' +
dia.substring(3,7);
      else
        dia = dia.substring(0,2) + '/' + dia.substring(2,4) + '/' +
dia.substring(4,8);

      AlgC.comunic.tEnviadas.append("Para " + r + ": Marcação de reunião para o
dia " + dia);
      if(Integer.parseInt(duracao)==1)
        AlgC.comunic.tEnviadas.append(" com  a duração de " + duracao + "
hora.\n" );
      else
        AlgC.comunic.tEnviadas.append(" com  a duração de " + duracao + "
horas.\n" );
    }

  }

  public void imprimeEmJComunic2(String conteudo)
  {
    String r = AlgC.proximoValor(conteudo);
    conteudo = AlgC.retiraValor(conteudo);

    String hora = AlgC.proximoValor(conteudo);
    conteudo = AlgC.retiraValor(conteudo);
    String disp = AlgC.proximoValor(conteudo);
    conteudo = AlgC.retiraValor(conteudo);
    String simNao = AlgC.proximoValor(conteudo);
    AlgC.comunic.tRecebidas.append("De " + r + ": Às " + hora + " tenho a
disponibilidade de " + disp + '.');

    if(simNao.equals("yes"))
    {
      conteudo = AlgC.retiraValor(conteudo);
      String altHora = AlgC.proximoValor(conteudo);
      conteudo = AlgC.retiraValor(conteudo);
      String altDisp = AlgC.proximoValor(conteudo);

      AlgC.comunic.tRecebidas.append(" Como alternativa às " + altHora + "
tenho a disponibilidade de " + altDisp + '.');
    }

    AlgC.comunic.tRecebidas.append("\n");
  }

  public void trataSolucao(String conteudo)
  {
    int solucao = Integer.parseInt(AlgC.proximoValor(conteudo));

    if(solucao != -1)
    {
      conteudo = AlgC.retiraValor(conteudo);
      float media = Float.parseFloat(AlgC.proximoValor(conteudo));
```

```
      AlgC.comunic.lSolucao2.setText("Reunião às " + solucao + " com um grau
médio de indisponibilidades de " + media + '.');
      AlgC.comunic.bOutraS.setEnabled(true);
      AlgC.comunic.bConfirmar.setEnabled(true);
    }
    else
    {
      AlgC.comunic.lSolucao2.setText("Esgotaram-se todas as hipóteses dentro
dos critérios pedidos!");
    }
  }
}
```

Ocup.java

```
import java.io.*;

class Ocup implements Serializable
{
  String descricao;
  int horai;
  int horaf;
  int prior;

  public Ocup(String desc,int hi, int hf, int p)
  {
    descricao = desc.toUpperCase();
    horai = hi;
    horaf = hf;
    prior = p;
  }

  public String devolveDescricao()
  {
    return descricao;
  }

  public int devolveHInicial()
  {
    return horai;
  }

  public int devolveHFinal()
  {
    return horaf;
  }

  public int devolvePrioridade()
  {
    return prior;
  }

  public int devolveIntervalo()
  {
    return (horaf-horai);
  }

  public boolean novaDescricao(String desc)
  {
    if(!desc.equals(""))
    {
      descricao = desc;
```

```
      return true;
    }
    return false;
  }

  public boolean horaInicial(int hi)
  {
    if(hi<horaf)
    {
      horai = hi;
      return true;
    }
    return false;
  }

  public boolean horaFinal(int hf)
  {
    if(hf>horai)
    {
      horaf = hf;
      return true;
    }
    return false;
  }

  public boolean novaPrioridade(int p)
  {
    if(p>=1 && p<6)
    {
      prior = p;
      return true;
    }
    return false;
  }

  public int devolveLinhaInicial()
  {
    return (horai-AlgC.HoraInicial);
  }

  public int devolveLinhaFinal()
  {
    return (horaf-AlgC.HoraInicial);
  }
}
```

JPass.java

```
import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JPasswordField;
import javax.swing.JButton;
import javax.swing.JPanel;
import java.beans.*; //Property change stuff
import java.awt.*;
import java.awt.event.*;

class JPass extends JDialog
{
  //própria janela para ser usado nas "inner classes".
```

```
private JDialog eu;

private JTextField tLogin;
private JTextField tLocal;
private JTextField tPorto;
private JPasswordField tPass;

public JPass()
{
  super();
  eu = this;

  this.setResizable(false);
  setTitle("Segurança - o acesso interdito é punido por lei!");

  //criação e inicialização dos labels da janela
  JLabel mLogin = new JLabel("Login:");
  JLabel mPass = new JLabel("Password:");
  JLabel mLocal = new JLabel("Localização do Servidor:");
  JLabel mPorto = new JLabel("Porto:");

  //inicialização dos campos de introdução dos dados
  tLogin = new JTextField(10);
  tLocal = new JTextField(15);
  tPorto = new JTextField(4);
  tPass = new JPasswordField(10);
  tPass.setEchoChar('*');

  //parte norte da janela
  JPanel painelN = new JPanel();
  JLabel lPass = new JLabel("Autenticação");
  lPass.setForeground(Color.black);
  Font font = new Font("Tahoma",Font.BOLD,15);
  lPass.setFont(font);
  painelN.add(lPass);

  //parte centro da janela
  JPanel painelC = new JPanel();
  painelC.setLayout(new GridLayout(4,2,0,5));
  painelC.add(mLogin);
  painelC.add(tLogin);
  painelC.add(mPass);
  painelC.add(tPass);
  painelC.add(mLocal);
  painelC.add(tLocal);
  painelC.add(mPorto);
  painelC.add(tPorto);

  //parte sul da janela
  JButton bEntrar = new JButton("Entrar");
  JButton bCancelar = new JButton("Cancelar");
  JPanel painelS = new JPanel();
  painelS.add(bEntrar);
  painelS.add(bCancelar);

  //disposição dos objectos
  JPanel painelPrincipal = new JPanel();
  painelPrincipal.setLayout(new BorderLayout(15,15));
  painelPrincipal.add(new JPanel(), BorderLayout.EAST);
  painelPrincipal.add(new JPanel(), BorderLayout.WEST);

  painelPrincipal.add(painelN, BorderLayout.NORTH);
  painelPrincipal.add(painelC, BorderLayout.CENTER);
  painelPrincipal.add(painelS, BorderLayout.SOUTH);
```

```
    setContentPane(painelPrincipal);


    //Acções dos butões e campos de texto

    bEntrar.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        String pass = new String(tPass.getPassword());
        String login = tLogin.getText();
        String local = tLocal.getText();
        String porto = tPorto.getText();

        if(login.equals(""))
          erro("É necessário introduzir o login.","Erro");
        else if(pass.equals(""))
               erro("É necessário introduzir a password","Erro");
        else if(local.equals(""))
               erro("É necessário introduzir o local do programa
servidor.","Erro");
        else if(porto.equals(""))
               erro("É necessário introduzir o porto do programa
servidor.","Erro");
        else if(!AgendaC.cliente.ligaAoServidor(local,porto))
               erro("Não é possivel estabelecer a ligação com o programa
servidor.","Erro");
        else if(!AgendaC.cliente.login(login,pass))
               erro("Não é possivel enviar mensagem KQML ao programa
servidor.","Erro");
        else
        {
          int resp = AlgC.esperaResposta();

          if(resp == 0)
          {
            AgendaC.cliente.logout();
            erro("Login ou Password Incorrectos!","O acesso não autorizado e
punido por lei!");
          }
          else
          {
            //password correcta - avança para a janela principal
            JAgenda agenda = new JAgenda(tLogin.getText());
            agenda.setSize(870,575);
            agenda.setLocation(70,66);
            tPass.setText("");

            //inicializa o horário
            AlgC.inicializaHorario();

            AlgC.esperaResposta();

            agenda.show();
            eu.dispose();
          }
        }
      }
    });

    bCancelar.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        System.exit(0);
```

```
      }
    });

    tLogin.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        tLogin.transferFocus();
      }
    });

    tPass.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        tPass.transferFocus();
      }
    });

    tLocal.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        tLocal.transferFocus();
      }
    });

    tPorto.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        tPorto.transferFocus();
      }
    });

    //para fechar o program ao carregar na cruz
    addWindowListener(new WindowAdapter()
    {
      public void windowClosing(WindowEvent e)
      {
        System.exit(0);
      }
    });
  }

  public void erro(String msg,String titulo)
  {
    JOptionPane.showMessageDialog(eu, msg, titulo, JOptionPane.ERROR_MESSAGE);
  }
}
```

TAgenda.java

```
import javax.swing.JTable;
import javax.swing.table.AbstractTableModel;
import java.awt.*;
import java.util.*;

//A classe principal da aplicação cliente

class TAgenda extends JTable
{
  int dia, mes, ano , d, m, a, dif, largura, altura, nletras;
  int npixeis = 8;
```

```
  LinkedList lista;
  ListIterator iLista;
  Ocup ocupacao;
  Integer chave;

  public TAgenda(AbstractTableModel modelo)
  {
    super(modelo);
  }

  public void paint(Graphics g)
  {
    super.paintComponent(g);

    d = AlgC.agora.get(AlgC.agora.DAY_OF_MONTH);
    m = AlgC.agora.get(AlgC.agora.MONTH);
    a = AlgC.agora.get(AlgC.agora.YEAR);

    Calendar cal = new GregorianCalendar(a,m,d);

    cal.add(cal.DATE,-7);

    for(int i=1;i<=7;i++)
    {
      cal.add(cal.DATE,+1);

      dia = cal.get(cal.DAY_OF_MONTH);
      mes = cal.get(cal.MONTH)+1;
      ano = cal.get(cal.YEAR);

      chave = AlgC.determinaChave(dia,mes,ano);

      lista = (LinkedList)AlgC.horario.get(chave);

      if(lista!=null)
      {
        iLista = lista.listIterator(0);

        while(iLista.hasNext())
        {
          ocupacao = (Ocup)iLista.next();
          //System.out.println(ocupacao.devolveDescricao());

pintaOcupacao(ocupacao.devolveDescricao(),ocupacao.devolveLinhaInicial(),ocupac
ao.devolveLinhaFinal(),i,g);
        }
      }
    }
    if(AlgC.seleccao!=null)

pintaSeleccao(AlgC.seleccao.devolveLinhaInicial(),AlgC.seleccao.devolveLinhaFin
al(),AlgC.colunaSeleccao,g);
  }

  private void pintaOcupacao(String descricao, int linhaInicial, int
linhaFinal, int coluna, Graphics g)
  {
    Rectangle rect = getCellRect(linhaInicial,coluna,true);
    dif = linhaFinal-linhaInicial;
    largura = (int)rect.getWidth();
    altura = dif * (int)rect.getHeight();
    nletras = largura/npixeis-1;
    if(nletras < descricao.length())
      descricao = descricao.substring(0,nletras);
```

```
    //pinta rectangulo da ocupacão
    g.setColor(Color.black);
    g.fillRect((int)rect.getX(),(int)rect.getY(),largura,altura);

    g.setColor(Color.white);
    g.drawRect((int)rect.getX(),(int)rect.getY(),largura,altura);

    //pinta descricao da ocupação

g.drawString(descricao,(int)rect.getX()+(nletras/2),(int)rect.getY()+(altura/2)
);
  }

  private void pintaSeleccao(int linhaInicial, int linhaFinal, int coluna,
Graphics g)
  {
    Rectangle rect = getCellRect(linhaInicial,coluna,true);
    dif = linhaFinal-linhaInicial;
    largura = (int)rect.getWidth();
    altura = dif * (int)rect.getHeight();

    //pinta rectangulo exterior da seleccao
    g.setColor(Color.red);
    g.fillRect((int)rect.getX(),(int)rect.getY(),largura,4);
    g.fillRect((int)rect.getX()+largura-4,(int)rect.getY(),4,altura);
    g.fillRect((int)rect.getX(),(int)rect.getY(),4,altura);
    g.fillRect((int)rect.getX(),(int)rect.getY()+altura-4,largura,4);
  }
}
```

TModelo.java

```
import javax.swing.table.AbstractTableModel;

//modelo da tabela
class TModelo extends AbstractTableModel
{
  //nomes das colunas
  final String[] nomeColuna = {"Horas",
                               "Segunda",
                               "Terça",
                               "Quarta",
                               "Quinta",
                               "Sexta",
                               "Sábado",
                               "Domingo"};

  final Object[][] dados = new Object[(AlgC.HoraFinal-AlgC.HoraInicial+1)][8];

  public int getColumnCount()
  {
    return nomeColuna.length;
  }

  public int getRowCount()
  {
    return dados.length;
  }

  public String getColumnName(int col)
  {
    return nomeColuna[col];
  }
```

```
  public Object getValueAt(int row, int col)
  {
    return dados[row][col];
  }

  public void setValueAt(Object value, int row, int col)
  {
    dados[row][col] = value;
  }
}
```

JAgenda.java

```
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.table.*;

//A janela principal da aplicacao cliente

class JAgenda extends JFrame
{
  //nome da janela principal
  private static final String Titulo = "Agenda 2000";

  //própria janela para ser usado nas "inner classes".
  private JFrame eu;

  //margens das toolbar's
  private Insets margemToolBar = new Insets(5,5,5,5);

  //menu principal de topo
  JMenuBar menu;

  //toolbar da parte norte da janela
  private JToolBar toolBarNorte;

  //toolbar da parte oeste da janela
  private JToolBar toolBarOeste;

  //tabela e respectivo scroll do centro da janela
  private TAgenda tabela;
  public static JScrollPane scrollpane;

  //botões
  private JButton butao;

  //Label da semana da parte de cima da janela
  private JLabel labelSemana;

  //resolucao do bug de clearSelection()
  private boolean flag = true;

  //definição das accções
  private Action novaLigacao;
  private Action sair;
  private Action abreJOcupacao;
  private Action apagaOcupacao;
  private Action abreJReuniao;
```

```
  private Action semanaSeguinte;
  private Action semanaAnterior;
  private Action acerca;

  //construtor
  public JAgenda(String login)
  {
    super(Titulo + "   Bemvindo - " + login);
    eu = this;

    //constroi accoes
    constroiAccoes();

    //constroi menu
    constroiMenu();

    //constroi parte norte da janela
    toolBarNorte = new JToolBar();
    toolBarNorte.setMargin(margemToolBar);
    constroiNorte();

    //constroi parte oeste da janela
    toolBarOeste = new JToolBar();
    toolBarOeste.setMargin(margemToolBar);
    constroiOeste();

    //constroi parte centro da janela
    constroiCentro();
    scrollpane = new JScrollPane(tabela);

    // Selecções feitas pelo rato
    tabela.addMouseListener(new rato());

    //painel principal
    JPanel painelPrincipal = new JPanel();
    painelPrincipal.setLayout(new BorderLayout());
    painelPrincipal.add(toolBarNorte, BorderLayout.NORTH);
    painelPrincipal.add(toolBarOeste,BorderLayout.WEST);
    painelPrincipal.add(scrollpane,BorderLayout.CENTER);
    painelPrincipal.add(new JPanel(),BorderLayout.EAST);
    painelPrincipal.add(new JPanel(),BorderLayout.SOUTH);

    setContentPane(painelPrincipal);

    //para fechar o program ao carregar na cruz
    this.addWindowListener(new WindowAdapter()
    {
      public void windowClosing(WindowEvent e)
      {
        AgendaC.cliente.logout();
        System.exit(0);
      }
    });
  }

  private void constroiAccoes()
  {
    novaLigacao = new AbstractAction("Ligar como Outro Utilizador", new
ImageIcon("imagens/liga.gif"))
    {
      public void actionPerformed(ActionEvent e)
      {
        //reinicializacao das variaveis
        AlgC.inicializaHorario();
        AlgC.seleccao=null;
```

```
        AlgC.semCache=0;
        AlgC.minCache=0;
        AlgC.maxCache=0;

        AlgC.agora = new GregorianCalendar();

        int diaSemana = AlgC.agora.get(Calendar.DAY_OF_WEEK);

        if(diaSemana==1)
          diaSemana = 8;

        AlgC.agora.add(AlgC.agora.DATE,-(diaSemana - 2));

        int diai = AlgC.agora.get(AlgC.agora.DAY_OF_MONTH);
        int mesi = AlgC.agora.get(AlgC.agora.MONTH)+1;
        int anoi = AlgC.agora.get(AlgC.agora.YEAR);

        AlgC.agora.add(AlgC.agora.DATE,+6);

        int diaf = AlgC.agora.get(Calendar.DAY_OF_MONTH);
        int mesf = AlgC.agora.get(Calendar.MONTH)+1;
        int anof = AlgC.agora.get(Calendar.YEAR);

        //label da semana actual com letra realçada
        labelSemana.setText("Horário da Semana: " +
                                    diai + '/' + mesi + '/' + anoi +
                                  " a " + diaf + '/' + mesf + '/' + anof);

        AgendaC.cliente.logout();
        AgendaC.janelaPass.show();
        eu.dispose();
      }
    };

    sair = new AbstractAction("Sair", new ImageIcon("imagens/sair.gif"))
    {
      public void actionPerformed(ActionEvent e)
      {
        AgendaC.cliente.logout();
        System.exit(0);
      }
    };

    abreJOcupacao = new AbstractAction("Nova Ocupação", new
ImageIcon("imagens/novo_bloco1.gif"))
    {
      public void actionPerformed(ActionEvent e)
      {
        int coluna;
        int[] linhas, data;
        JOcup ocup;

        if((coluna=tabela.getSelectedColumn())>0 && flag)
        {
          //calculo do dia seleccionado
          data = AlgC.dataColuna(coluna);

          if(AlgC.seleccao==null)
          {
            //janela com dados das horas
            //calculo das horas seleccionadas
            linhas = AlgC.horasLinha(tabela.getSelectedRows());

            //janela com dados
            ocup = new JOcup(eu,data[0],data[1],data[2],linhas[0],linhas[1]);
```

```
          }
          else
          {
            //janela com dados da seleccao
            ocup = new JOcup(eu,data[0],data[1],data[2],AlgC.seleccao);
          }
        }
        else
        {
          //janela sem dados
          ocup = new JOcup(eu);
        }
        //ocup.pack();
        ocup.setSize(430,290);
        ocup.setLocationRelativeTo(eu);
        ocup.show();
      }
    };

    apagaOcupacao = new AbstractAction("Apaga Ocupação", new
ImageIcon("imagens/apagar_bloco1.gif"))
    {
      public void actionPerformed(ActionEvent e)
      {
        if(AlgC.seleccao!=null)
        {
          int[] data = AlgC.dataColuna(AlgC.colunaSeleccao);
          String descricao = AlgC.seleccao.devolveDescricao();
          int horai = AlgC.seleccao.devolveHInicial();
          int horaf = AlgC.seleccao.devolveHFinal();
          int prior = AlgC.seleccao.devolvePrioridade();


if(AgendaC.cliente.apagaOcupacao(data[0]+","+data[1]+","+data[2]+","+descricao+
","+horai+","+horaf+","+prior))
          {
            AlgC.apagaOcupacao(data[0],data[1],data[2],AlgC.seleccao);
            AlgC.seleccao = null;
          }
          else
          {
            erro("Não é possivel estabelecer a ligação com o programa servidor
(P.F. Tente ligar o programa de novo).");
            System.exit(0);
          }
        }
      }
    };

    abreJReuniao = new AbstractAction("Marcar Nova Reunião", new
ImageIcon("imagens/reuniao1.gif"))
    {
      public void actionPerformed(ActionEvent e)
      {
        JReuniao reuniao = new JReuniao(eu);
        //reuniao.pack();
        reuniao.setSize(600,450);
        reuniao.setLocationRelativeTo(eu);
        reuniao.show();
      }
    };

    semanaAnterior = new AbstractAction("Semana Anterior", new
ImageIcon("imagens/anterior.gif"))
    {
```

```
      public void actionPerformed(ActionEvent e)
      {
        AlgC.semCache--;

        //só pede semana ao servidor se for necessário
        if(AlgC.semCache<AlgC.minCache)
        {
          AgendaC.cliente.pedeSemana(AlgC.semCache);
          AlgC.minCache--;
          AlgC.esperaResposta();
        }

        AlgC.agora.add(AlgC.agora.DATE,-13);
        int diai = AlgC.agora.get(AlgC.agora.DAY_OF_MONTH);
        int mesi = AlgC.agora.get(AlgC.agora.MONTH)+1;
        int anoi = AlgC.agora.get(AlgC.agora.YEAR);

        AlgC.agora.add(AlgC.agora.DATE,+6);
        int diaf = AlgC.agora.get(AlgC.agora.DAY_OF_MONTH);
        int mesf = AlgC.agora.get(AlgC.agora.MONTH)+1;
        int anof = AlgC.agora.get(AlgC.agora.YEAR);

        labelSemana.setText("Horário da Semana: " +
                            diai + '/' + mesi + '/' + anoi +
                            " a " + diaf + '/' + mesf + '/' + anof);
        tabela.clearSelection();
        flag = false;
        AlgC.seleccao = null;
        scrollpane.repaint();
      }
    };

    semanaSeguinte = new AbstractAction("SemanaSeguinte", new
ImageIcon("imagens/seguinte.gif"))
    {
      public void actionPerformed(ActionEvent e)
      {
        AlgC.semCache++;

        //só pede semana ao servidor se for necessário
        if(AlgC.semCache>AlgC.maxCache)
        {
          AgendaC.cliente.pedeSemana(AlgC.semCache);
          AlgC.maxCache++;
          AlgC.esperaResposta();
        }

        AlgC.agora.add(AlgC.agora.DATE,+1);
        int diai = AlgC.agora.get(AlgC.agora.DAY_OF_MONTH);
        int mesi = AlgC.agora.get(AlgC.agora.MONTH)+1;
        int anoi = AlgC.agora.get(AlgC.agora.YEAR);

        AlgC.agora.add(AlgC.agora.DATE,+6);
        int diaf = AlgC.agora.get(AlgC.agora.DAY_OF_MONTH);
        int mesf = AlgC.agora.get(AlgC.agora.MONTH)+1;
        int anof = AlgC.agora.get(AlgC.agora.YEAR);

        labelSemana.setText("Horário da Semana: " +
                            diai + '/' + mesi + '/' + anoi +
                            " a " + diaf + '/' + mesf + '/' + anof);
        tabela.clearSelection();
        flag = false;
        AlgC.seleccao = null;
        scrollpane.repaint();
      }
```

```
    };

    acerca = new AbstractAction("Acerca da Agenda 2000...", new
ImageIcon("imagens/acerca.gif"))
    {
      public void actionPerformed(ActionEvent e)
      {
        JAcerca acerca = new JAcerca(eu);
        acerca.setLocationRelativeTo(eu);
        acerca.show();
      }
    };
  }

  private void constroiMenu()
  {
    // Constroi o menu de topo
    menu = new JMenuBar();

    // Menu "Agenda"
    JMenu mAgenda = new JMenu("Agenda");
    mAgenda.setMnemonic(KeyEvent.VK_A);
    mAgenda.add(novaLigacao);
    mAgenda.addSeparator();
    mAgenda.add(sair);

    JMenu mHorario = new JMenu("Horário");
    mHorario.setMnemonic(KeyEvent.VK_H);
    mHorario.add(abreJOcupacao);
    mHorario.add(apagaOcupacao);
    mHorario.addSeparator();
    mHorario.add(abreJReuniao);

    JMenu mSemana = new JMenu("Semana");
    mSemana.setMnemonic(KeyEvent.VK_S);
    mSemana.add(semanaSeguinte);
    mSemana.add(semanaAnterior);

    JMenu mAcerca = new JMenu("Acerca");
    mAcerca.setMnemonic(KeyEvent.VK_C);
    mAcerca.add(acerca);

    menu.add(mAgenda);
    menu.add(mHorario);
    menu.add(mSemana);
    menu.add(mAcerca);
    setJMenuBar(menu);
  }

  private void constroiNorte()
  {
    //não se pode fazer drag and drop
    toolBarNorte.setFloatable(false);
    //para centrar
    toolBarNorte.addSeparator(new Dimension(100,1));

    int diaSemana = AlgC.agora.get(Calendar.DAY_OF_WEEK);

    if(diaSemana==1)
      diaSemana = 8;

    AlgC.agora.add(AlgC.agora.DATE,-(diaSemana - 2));

    int diai = AlgC.agora.get(AlgC.agora.DAY_OF_MONTH);
    int mesi = AlgC.agora.get(AlgC.agora.MONTH)+1;
```

```
  int anoi = AlgC.agora.get(AlgC.agora.YEAR);

  AlgC.agora.add(AlgC.agora.DATE,+6);

  int diaf = AlgC.agora.get(Calendar.DAY_OF_MONTH);
  int mesf = AlgC.agora.get(Calendar.MONTH)+1;
  int anof = AlgC.agora.get(Calendar.YEAR);

  //label da semana actual com letra realçada
  labelSemana = new JLabel("Horário da Semana: " +
                                  diai + '/' + mesi + '/' + anoi +
                                 " a " + diaf + '/' + mesf + '/' + anof);
  labelSemana.setForeground(Color.black);
  Font font = new Font("Tahoma",Font.BOLD,22);
  labelSemana.setFont(font);
  labelSemana.setAlignmentY(CENTER_ALIGNMENT);
  toolBarNorte.add(labelSemana);

  JPanel painel = new JPanel(new FlowLayout(FlowLayout.RIGHT));

  //botao semana anterior
  butao = new JButton(new ImageIcon("imagens/anterior.gif"));
  butao.setToolTipText("Semana Anterior");
  butao.setAlignmentY(CENTER_ALIGNMENT);
  butao.addActionListener(semanaAnterior);
  painel.add(butao);

  //botao semana seguinte
  butao = new JButton(new ImageIcon("imagens/seguinte.gif"));
  butao.setToolTipText("Semana Seguinte");
  butao.setAlignmentY(CENTER_ALIGNMENT);
  butao.addActionListener(semanaSeguinte);
  painel.add(butao);
  toolBarNorte.add(painel);
}

private void constroiOeste()
{
  //não se pode fazer drag and drop
  toolBarOeste.setFloatable(false);
  toolBarOeste.setOrientation(JToolBar.VERTICAL);

  //separador
  toolBarOeste.addSeparator(new Dimension(1,30));

  //butão nova ocupação
  butao = new JButton(new ImageIcon("imagens/novo_bloco.gif"));
  butao.setToolTipText("Introduzir nova ocupação");
  butao.addActionListener(abreJOcupacao);
  toolBarOeste.add(butao);

  //separador
  toolBarOeste.addSeparator();

  //butão apagar ocupação
  butao = new JButton(new ImageIcon("imagens/apagar_bloco.gif"));
  butao.setToolTipText("Apagar ocupação");
  butao.addActionListener(apagaOcupacao);
  toolBarOeste.add(butao);

  //separador
  toolBarOeste.addSeparator(new Dimension(1,30));

  //butão marcar reunião
  butao = new JButton(new ImageIcon("imagens/reuniao.gif"));
```

```
  butao.setToolTipText("Marcar nova reunião");
  butao.addActionListener(abreJReuniao);
  toolBarOeste.add(butao);
}

private void constroiCentro()
{

  TModelo modelo = new TModelo();
  tabela = new TAgenda(modelo);

  //tamanho das colunas
  tabela.setRowHeight(AlgC.TamLinhas/(AlgC.HoraFinal-AlgC.HoraInicial));
  tabela.setGridColor(Color.black);
  tabela.setSelectionForeground(Color.red);
  tabela.setSelectionBackground(new Color(198,39,39));
  tabela.setRowSelectionAllowed(false);
  tabela.setCellSelectionEnabled(true);
  tabela.setLayout(new FlowLayout(FlowLayout.CENTER));


  TableColumn column = tabela.getColumnModel().getColumn(0);
  column.setPreferredWidth(25);

  //coloca as horas
  for(int i=AlgC.HoraInicial;i<=AlgC.HoraFinal;i++)
    tabela.setValueAt(i+":00",i-AlgC.HoraInicial,0);
}

public void erro(String msg)
{
  JOptionPane.showMessageDialog(eu, msg,"Erro",JOptionPane.ERROR_MESSAGE);
}

public static void main(String[] argv)
{
  JAgenda agenda = new JAgenda("ei97007");
  agenda.setSize(870,575);
  agenda.setLocation(70,66);
  agenda.show();
}

class rato extends MouseAdapter
{
  public void mousePressed(MouseEvent e)
  {
    AlgC.seleccao = null;
    flag = true;
  }

  public void mouseReleased(MouseEvent e)
  {
    int coluna;
    int[] linhas, data;

    if((coluna=tabela.getSelectedColumn())>0)
    {
      data = AlgC.dataColuna(coluna);

      linhas = AlgC.horasLinha(tabela.getSelectedRows());

      if(!AlgC.testaSeleccao(data[0],data[1],data[2],linhas[0],linhas[1]))
      {
         AlgC.seleccao = AlgC.conflito;
         AlgC.colunaSeleccao = coluna;
```

```
            tabela.clearSelection();
          }
        }
      }
    }
  }
}
```

JOcup.java

```
import javax.swing.*;
import java.beans.*; //Property change stuff
import java.awt.*;
import java.awt.event.*;
import java.util.*;

class JOcup extends JDialog
{
  //própria janela para ser usado nas "inner classes".
  private JDialog eu;

  private JTextField tDescricao;
  private JTextField tDia;
  private JTextField tMes;
  private JTextField tAno;
  private JComboBox cHoraInicio;
  private JComboBox cHoraFim;
  private JRadioButton[] rbPref = new JRadioButton[5];
  private ButtonGroup bgGrupo;

  private Action bOKAccao;

  public JOcup(Frame janela)
  {
    super(janela, true);
    eu = this;

    constroiJanela();
  }

  public JOcup(Frame janela, int dia, int mes, int ano, int horai, int horaf)
  {
    super(janela, true);
    eu = this;

    constroiJanela();

    tDia.setText((new Integer(dia)).toString());
    tMes.setText((new Integer(mes)).toString());
    tAno.setText((new Integer(ano)).toString());
    cHoraInicio.setSelectedIndex((horai-AlgC.HoraInicial));
    cHoraFim.setSelectedIndex((horaf-(AlgC.HoraInicial+1)));
 }

  public JOcup(Frame janela, int dia, int mes, int ano, Ocup ocupacao)
  {
    super(janela, true);
    eu = this;

    constroiJanela();

    tDia.setText((new Integer(dia)).toString());
    tMes.setText((new Integer(mes)).toString());
```

```
    tAno.setText((new Integer(ano)).toString());
    tDescricao.setText(ocupacao.devolveDescricao());
    cHoraInicio.setSelectedIndex((ocupacao.devolveHInicial()-
AlgC.HoraInicial));
    cHoraFim.setSelectedIndex((ocupacao.devolveHFinal()-(AlgC.HoraInicial+1)));
    rbPref[(ocupacao.devolvePrioridade()-1)].setSelected(true);
  }

  private void constroiJanela()
  {
    setTitle("Ocupação");
    setResizable(false);

    //parte norte da janela
    JPanel painelN = new JPanel();
    JLabel lOcup = new JLabel("Ocupação");
    lOcup.setForeground(Color.black);
    Font font = new Font("Tahoma",Font.BOLD,15);
    lOcup.setFont(font);
    painelN.add(lOcup);

    //parte centro da janela
    JPanel painelC = new JPanel(new GridLayout(4,1));
    JPanel painelC1 = new JPanel(new FlowLayout(FlowLayout.LEFT));
    JPanel painelC2 = new JPanel(new FlowLayout(FlowLayout.LEFT));
    JPanel painelC3 = new JPanel(new FlowLayout(FlowLayout.LEFT));
    JPanel painelC4 = new JPanel(new FlowLayout(FlowLayout.LEFT));

    //criação e inicialização dos labels da janela
    JLabel lDescricao = new JLabel("Descrição:    ");
    JLabel lData = new JLabel("Data: ");
    JLabel lHoraInicio = new JLabel("Hora Início: ");
    JLabel lHoraFim = new JLabel("Hora Fim:    ");

    //inicialização dos campos de introdução dos dados
    String[] horasI = new String[(AlgC.HoraFinal-AlgC.HoraInicial+1)];
    String[] horasF = new String[(AlgC.HoraFinal-AlgC.HoraInicial+1)];

    for(int i=AlgC.HoraInicial; i<=AlgC.HoraFinal; i++)
    {
      horasI[i-AlgC.HoraInicial] = i + ":00";
      horasF[i-AlgC.HoraInicial] = (i+1) + ":00";
    }

    tDescricao = new JTextField(20);
    tDia = new JTextField(2);
    tMes = new JTextField(2);
    tAno = new JTextField(4);
    cHoraInicio = new JComboBox(horasI);
    cHoraFim = new JComboBox(horasF);

    //localização dos componentes
    painelC1.add(lDescricao);
    painelC1.add(tDescricao);
    painelC2.add(lData);
    painelC2.add(tDia);
    painelC2.add(new JLabel("/"));
    painelC2.add(tMes);
    painelC2.add(new JLabel("/"));
    painelC2.add(tAno);
    painelC3.add(lHoraInicio);
    painelC3.add(cHoraInicio);
    painelC3.add(lHoraFim);
    painelC3.add(cHoraFim);
```

```
//criação dos radio buttons
rbPref[0] = new JRadioButton("Min.");
rbPref[1] = new JRadioButton("Pouca");
rbPref[2] = new JRadioButton("Média");
rbPref[3] = new JRadioButton("Muita");
rbPref[4] = new JRadioButton("Max.");

//valores dos radio buttons
rbPref[0].setActionCommand("1");
rbPref[1].setActionCommand("2");
rbPref[2].setActionCommand("3");
rbPref[3].setActionCommand("4");
rbPref[4].setActionCommand("5");

//valor por defeito
rbPref[2].setSelected(true);

//grupo de radio buttons
bgGrupo = new ButtonGroup();
bgGrupo.add(rbPref[0]);
bgGrupo.add(rbPref[1]);
bgGrupo.add(rbPref[2]);
bgGrupo.add(rbPref[3]);
bgGrupo.add(rbPref[4]);

JLabel lPrior = new JLabel("Prioridade:");
painelC4.add(lPrior);

painelC4.add(rbPref[0]);
painelC4.add(rbPref[1]);
painelC4.add(rbPref[2]);
painelC4.add(rbPref[3]);
painelC4.add(rbPref[4]);

painelC.add(painelC1);
painelC.add(painelC2);
painelC.add(painelC3);
painelC.add(painelC4);

//parte sul da janela
JPanel painelS = new JPanel();

//botões ok e cancelar
JButton bOK = new JButton("OK");
constroiAccao();
bOK.addActionListener(bOKAccao);
JButton bCancelar = new JButton("Cancelar");

painelS.add(bOK);
painelS.add(bCancelar);

//disposição dos objectos
JPanel painelPrincipal = new JPanel();
painelPrincipal.setLayout(new BorderLayout(15,15));
painelPrincipal.add(new JPanel(), BorderLayout.EAST);
painelPrincipal.add(new JPanel(), BorderLayout.WEST);

painelPrincipal.add(painelN, BorderLayout.NORTH);
painelPrincipal.add(painelC, BorderLayout.CENTER);
painelPrincipal.add(painelS, BorderLayout.SOUTH);

this.setContentPane(painelPrincipal);

//Acções dos butões e campos de texto
bCancelar.addActionListener(new ActionListener()
```

```
    {
      public void actionPerformed(ActionEvent e)
      {
        eu.dispose();
      }
    });

    tDescricao.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        tDescricao.transferFocus();
      }
    });

    //para fechar o program ao carregar na cruz
    addWindowListener(new WindowAdapter()
    {
      public void windowClosing(WindowEvent e)
      {
        eu.dispose();
      }
    });
  }

  private void constroiAccao()
  {
    bOKAccao = new AbstractAction("OK")
    {
      public void actionPerformed(ActionEvent e)
      {


    int dia, mes, ano, horai, horaf, prior=0;
    String descricao;

    try
    {
      for(int i=0;i<5;i++)
      {
      if(rbPref[i].isSelected())
        {
          prior = i+1;
          break;
        }
      }

      descricao = tDescricao.getText();
      dia = Integer.parseInt(tDia.getText());
      mes = Integer.parseInt(tMes.getText());
      ano = Integer.parseInt(tAno.getText());
      horai = cHoraInicio.getSelectedIndex()+AlgC.HoraInicial;
      horaf = cHoraFim.getSelectedIndex()+AlgC.HoraInicial+1;

      if(descricao.equals(""))
        erro("Erro na introdução da Descição: a Descrição tem que ser
preenchida.");
      else if(!AlgC.verificaData(dia,mes,ano))
        erro("A data deve ter o formato dd/mm/yyyy e " +
             "tem de ser compreendida entre 1990 e 2010");
      else if(horaf<=horai)
        erro("A hora inicial não pode ser maior que a hora final.");
      else if(AlgC.seleccao!=null)
      {
        int[] data = AlgC.dataColuna(AlgC.colunaSeleccao);
```

```
        String d = AlgC.seleccao.devolveDescricao();
        int hi = AlgC.seleccao.devolveHInicial();
        int hf = AlgC.seleccao.devolveHFinal();
        int p = AlgC.seleccao.devolvePrioridade();


if(AgendaC.cliente.apagaOcupacao(data[0]+","+data[1]+","+data[2]+","+d+","+hi+"
,"+hf+","+p))
        {
          AlgC.apagaOcupacao(data[0],data[1],data[2],AlgC.seleccao);
        }
        else
        {
          erro("Não é possivel estabelecer a ligação com o programa servidor
(P.F. Tente ligar o programa de novo).");
          System.exit(0);
        }


if(AgendaC.cliente.novaOcupacao(dia+","+mes+","+ano+","+descricao+","+horai+","
+horaf+","+prior))
        {
          Ocup ocupacao = new Ocup(descricao,horai,horaf,prior);
          AlgC.insereOcupacao(dia,mes,ano,ocupacao);
          JAgenda.scrollpane.repaint();
          eu.dispose();
        }
        else
        {
          erro("Não é possivel estabelecer a ligação com o programa servidor
(P.F. Tente ligar o programa de novo).");
          System.exit(0);
        }
      }
      else
      {
        if(!AlgC.testaSeleccao(dia,mes,ano,horai,horaf))
        {
          erro("O intervalo escolhido entra em conflito com outra ocupação.");
        }
        else
        {

if(AgendaC.cliente.novaOcupacao(dia+","+mes+","+ano+","+descricao+","+horai+","
+horaf+","+prior))
          {
            Ocup ocupacao = new Ocup(descricao,horai,horaf,prior);
            AlgC.insereOcupacao(dia,mes,ano,ocupacao);
            JAgenda.scrollpane.repaint();
            eu.dispose();
          }
          else
          {
            erro("Não é possivel estabelecer a ligação com o programa servidor
(P.F. Tente ligar o programa de novo).");
            System.exit(0);
          }
        }
      }
    }
    catch(NumberFormatException l)
    {
      erro("Data mal introduzida. Esta deve ter o formato dd/mm/yyyy.");
    }
   }};
```

```
  }

  public void erro(String msg)
  {
    JOptionPane.showMessageDialog(eu, msg,"Erro",JOptionPane.ERROR_MESSAGE);
  }
}
```

JReuniao.java

```
import javax.swing.*;
import java.beans.*; //Property change stuff
import java.awt.*;
import java.awt.event.*;

class JReuniao extends JDialog
{
  //própria janela para ser usado nas "inner classes".
  private JDialog eu;

  private JTextField tDescricao;
  private JTextField tDiaInicio;
  private JTextField tMesInicio;
  private JTextField tAnoInicio;
  private JComboBox cHoraInicio;
  private JTextField tCriterio;
  private JComboBox cDuracao;
  private JList listaOutros1;
  private JList listaOutros2;
  private Object[] lista1;
  private Object[] lista2;

  public JReuniao(Frame janela)
  {
    super(janela, true);
    eu = this;

    setTitle("Nova Reunião");
    setResizable(false);

    //parte norte da janela
    JPanel painelN = new JPanel();
    JLabel lOcup = new JLabel("Marcação de Nova Reunião");
    lOcup.setForeground(Color.black);
    Font font = new Font("Tahoma",Font.BOLD,15);
    lOcup.setFont(font);
    painelN.add(lOcup);

    //parte centro da janela
    JPanel painelC = new JPanel(new GridLayout(2,1));
    JPanel painelC1 = new JPanel(new GridLayout(4,1));
    JPanel painelC11 = new JPanel(new FlowLayout(FlowLayout.LEFT,8,0));
    JPanel painelC12 = new JPanel(new FlowLayout(FlowLayout.LEFT,8,0));
    JPanel painelC13 = new JPanel(new FlowLayout(FlowLayout.LEFT,8,0));
    JPanel painelC14 = new JPanel(new GridLayout(1,3));
    JPanel painelC2 = new JPanel(new GridLayout(1,3));
    JPanel painelC21 = new JPanel(new GridLayout(2,1));
    JPanel painelC211 = new JPanel();
    JPanel painelC212 = new JPanel();

    //criação e inicialização dos labels da janela
    JLabel lDescricao = new JLabel("Descrição:    ");
    JLabel lDataInicio = new JLabel("Data Prevista: ");
    JLabel lHoraInicio = new JLabel("     Hora Prevista:  ");
```

```
JLabel lCriterio = new JLabel("Média máxima de Indisponibilidades:");
JLabel lDuracao = new JLabel("Duração:     ");
JLabel lOutros = new JLabel("Outras Agendas:");
JLabel lMarcarCom = new JLabel("Marcar Reunião com:");

//inicialização dos campos de introdução dos dados
String[] horas = new String[(AlgC.HoraFinal-AlgC.HoraInicial+1)];
String[] horasd = new String[(AlgC.HoraFinal-(AlgC.HoraInicial+1))];

for(int i=AlgC.HoraInicial; i<=AlgC.HoraFinal; i++)
{
  horas[i-AlgC.HoraInicial] = i + ":00";
}

for(int i=1; i<=(AlgC.HoraFinal-(AlgC.HoraInicial+1)); i++)
{
  horasd[i-1] = i + ":00";
}

tDescricao = new JTextField(25);
tDiaInicio = new JTextField(2);
tMesInicio = new JTextField(2);
tAnoInicio = new JTextField(4);
cHoraInicio = new JComboBox(horas);
tCriterio = new JTextField("1.00",3);
cDuracao = new JComboBox(horasd);

//localização dos componentes
painelC11.add(lDescricao);
painelC11.add(tDescricao);
painelC11.add(lDuracao);
painelC11.add(cDuracao);
painelC12.add(lDataInicio);
painelC12.add(tDiaInicio);
painelC12.add(new JLabel("/"));
painelC12.add(tMesInicio);
painelC12.add(new JLabel("/"));
painelC12.add(tAnoInicio);
painelC12.add(lHoraInicio);
painelC12.add(cHoraInicio);
painelC13.add(lCriterio);
painelC13.add(tCriterio);
painelC14.add(lOutros);
painelC14.add(new JPanel());
painelC14.add(lMarcarCom);

painelC1.add(painelC11);
painelC1.add(painelC12);
painelC1.add(painelC13);
painelC1.add(painelC14);

JButton bAdic = new JButton(new ImageIcon("imagens/seguinte.gif"));
bAdic.setToolTipText("Adicionar");
//bAdic.addActionListener(new semanaAnterior());

JButton bSub = new JButton(new ImageIcon("imagens/anterior.gif"));
bSub.setToolTipText("Remover");
//bSub.addActionListener(new semanaAnterior());

painelC211.add(bAdic);
painelC212.add(bSub);
painelC21.add(painelC211);
painelC21.add(painelC212);

pedeLista();
```

```
    lista2 = new Object[0];

    listaOutros1 = new JList(lista1);

listaOutros1.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
    JScrollPane listaOutrosScroll1 = new JScrollPane(listaOutros1);

    listaOutros2 = new JList();

listaOutros2.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
    JScrollPane listaOutrosScroll2 = new JScrollPane(listaOutros2);

    painelC2.add(listaOutrosScroll1);
    painelC2.add(painelC21);
    painelC2.add(listaOutrosScroll2);

    painelC.add(painelC1);
    painelC.add(painelC2);

    //parte sul da janela
    JPanel painelS = new JPanel();

    //botões marcar e cancelar
    JButton bMarcar = new JButton("Marcar");
    JButton bCancelar = new JButton("Cancelar");

    painelS.add(bMarcar);
    painelS.add(bCancelar);

    //disposição dos objectos
    JPanel painelPrincipal = new JPanel();
    painelPrincipal.setLayout(new BorderLayout(15,15));
    painelPrincipal.add(new JPanel(), BorderLayout.EAST);
    painelPrincipal.add(new JPanel(), BorderLayout.WEST);

    painelPrincipal.add(painelN, BorderLayout.NORTH);
    painelPrincipal.add(painelC, BorderLayout.CENTER);
    painelPrincipal.add(painelS, BorderLayout.SOUTH);

    setContentPane(painelPrincipal);


    //Acções dos butões e campos de texto

    bMarcar.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        int diai, mesi, anoi;
        float c;

        try
        {
          diai = Integer.parseInt(tDiaInicio.getText());
          mesi = Integer.parseInt(tMesInicio.getText());
          anoi = Integer.parseInt(tAnoInicio.getText());
          c = Float.parseFloat(tCriterio.getText());

          if(tDescricao.getText().equals(""))
            erro("Erro na introdução da Descição: a Descrição tem que ser
preenchida.");
          else if(!AlgC.verificaData(diai,mesi,anoi))
                 erro("A data deve ter o formato dd/mm/yyyy e " +
                  "tem de ser compreendida entre 1999 e 2010");
          else if(tCriterio.getText().equals(""))
```

```
                erro("Erro na introdução da média máxima de
indisponibilidades," +
                 " esta tem que ser preenchida.");
          else if(c<0 || c>5)
                erro("Erro na introdução da média máxima de
indisponibilidades, o valor tem de estar compreendido entre 0 e 5.");
          else if(lista2.length==0)
                 erro("É necessário indicar as pessoas com quem será feita a
marcação.");
          else
          {
            String data = new String();

            Integer chave = AlgC.determinaChave(diai, mesi, anoi);
            int hora = cHoraInicio.getSelectedIndex() + AlgC.HoraInicial;
            int duracao = cDuracao.getSelectedIndex()+1;

            data = chave.toString() + ',' + hora + ',' + duracao;
            AlgC.comunic = new JComunic(tDescricao.getText(), data , lista2 ,
c);
            //comunic.pack();
            AlgC.comunic.setResizable(false);
            AlgC.comunic.setSize(700,500);
            AlgC.comunic.setLocationRelativeTo(eu);
            AlgC.comunic.show();
            eu.dispose();
          }
        }
        catch(NumberFormatException l)
        {
          erro("Data inicial ou média máxima de indisponibilidades mal
introduzida. A data deve ter o formato dd/mm/yyyy.");
        }
      }
    });

    bCancelar.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        eu.dispose();
      }
    });

    bAdic.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        Object[] selec = listaOutros1.getSelectedValues();

        lista1 = AlgC.subtraiListas(lista1, selec);
        listaOutros1.setListData(lista1);

        lista2 = AlgC.ordenaListas(selec, lista2);
        listaOutros2.setListData(lista2);
      }
    });

    bSub.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        Object[] selec = listaOutros2.getSelectedValues();

        lista2 = AlgC.subtraiListas(lista2, selec);
```

```
      listaOutros2.setListData(lista2);

      lista1 = AlgC.ordenaListas(selec, lista1);
      listaOutros1.setListData(lista1);
    }
  });

  tDescricao.addActionListener(new ActionListener()
  {
    public void actionPerformed(ActionEvent e)
    {
      tDescricao.transferFocus();
    }
  });

  tDiaInicio.addActionListener(new ActionListener()
  {
    public void actionPerformed(ActionEvent e)
    {
      tDiaInicio.transferFocus();
    }
  });

  tMesInicio.addActionListener(new ActionListener()
  {
    public void actionPerformed(ActionEvent e)
    {
      tMesInicio.transferFocus();
    }
  });

  tAnoInicio.addActionListener(new ActionListener()
  {
    public void actionPerformed(ActionEvent e)
    {
      tAnoInicio.transferFocus();
    }
  });

  tCriterio.addActionListener(new ActionListener()
  {
    public void actionPerformed(ActionEvent e)
    {
      tCriterio.transferFocus();
    }
  });

  //para fechar o program ao carregar na cruz
  addWindowListener(new WindowAdapter()
  {
    public void windowClosing(WindowEvent e)
    {
      eu.dispose();
    }
  });
}

public void erro(String msg)
{
  JOptionPane.showMessageDialog(eu, msg,"Erro",JOptionPane.ERROR_MESSAGE);
}

public void pedeLista()
{
  int n=0;
```

```
    String utilAux;
    String util = AgendaC.cliente.pedeListaUtil();

    utilAux = util;
    while(utilAux.indexOf(",")!=-1)
    {
      utilAux = AlgC.retiraValor(utilAux);
      n++;
    }

    lista1 = new Object[n+1];
    n = 0;

    while(util.indexOf(",")!=-1)
    {
      lista1[n] = AlgC.proximoValor(util);
      util = AlgC.retiraValor(util);
      n++;
    }
    lista1[n] = AlgC.proximoValor(util);
  }

  public static void main(String[] argv)
  {
    JReuniao agenda = new JReuniao(new JFrame());
    agenda.setSize(870,575);
    agenda.setLocation(70,66);
    agenda.show();
  }
}
```

JComunic.java

```
import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JLabel;
import javax.swing.JTextArea;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import java.beans.*; //Property change stuff
import java.awt.*;
import java.awt.event.*;

class JComunic extends JDialog
{
  //própria janela para ser usado nas "inner classes".
  private JDialog eu;

  public JTextArea tEnviadas;
  public JTextArea tRecebidas;
  public JScrollPane sEnviadas;
  public JScrollPane sRecebidas;
  public JLabel lSolucao2;
  public JButton bOutraS;
  public JButton bConfirmar;
  public String mensagem;
  public String d;

  public JComunic(String msg, String data, Object[] lista, float c)
  {
    super();
    eu = this;
    mensagem = msg;
```

```
d = data;

setTitle("Comunicações");

//criação e inicialização dos labels da janela
JLabel lEnviadas = new JLabel("Mensagens enviadas:");
JLabel lRecebidas = new JLabel("Mensagens recebidas:");
JLabel lSolucao1 = new JLabel("Solução: ");
lSolucao2 = new JLabel("Pesquisando Solução...");

//inicialização dos campos de introdução dos dados
tEnviadas = new JTextArea();
tRecebidas = new JTextArea();

//não editáveis
tEnviadas.setEditable(false);
tRecebidas.setEditable(false);

//parte norte da janela
JPanel painelN = new JPanel(new FlowLayout(FlowLayout.CENTER,15,15));
JLabel lComunic = new JLabel("Comunicações");
lComunic.setForeground(Color.black);
Font font = new Font("Tahoma",Font.BOLD,15);
lComunic.setFont(font);
painelN.add(lComunic);

//parte centro da janela
JPanel painelC = new JPanel(new GridLayout(2,1,20,20));
JPanel painelC1 = new JPanel(new BorderLayout());
JPanel painelC2 = new JPanel(new BorderLayout());

sEnviadas = new JScrollPane(tEnviadas);
sRecebidas = new JScrollPane(tRecebidas);

painelC1.add(lEnviadas,BorderLayout.NORTH);
painelC1.add(sEnviadas,BorderLayout.CENTER);

painelC2.add(lRecebidas,BorderLayout.NORTH);
painelC2.add(sRecebidas,BorderLayout.CENTER);

painelC.add(painelC1);
painelC.add(painelC2);

//parte sul da janela
JPanel painelS = new JPanel(new GridLayout(2,1,10,10));
JPanel painelS1 = new JPanel(new FlowLayout(FlowLayout.LEFT,10,10));
JPanel painelS2 = new JPanel(new FlowLayout(FlowLayout.RIGHT,10,10));


font = new Font("Tahoma",Font.PLAIN,14);
lSolucao1.setFont(font);
lSolucao2.setFont(font);
lSolucao1.setForeground(Color.black);
lSolucao2.setForeground(Color.black);

painelS1.add(lSolucao1);
painelS1.add(lSolucao2);

bOutraS = new JButton("Outra Solução");
bConfirmar = new JButton("Confirmar Marcação");
JButton bCancelar = new JButton("Cancelar");
bOutraS.setEnabled(false);
bConfirmar.setEnabled(false);
painelS2.add(bOutraS);
painelS2.add(bConfirmar);
```

```
    painelS2.add(bCancelar);

    painelS.add(painelS1);
    painelS.add(painelS2);

    //disposição dos objectos
    JPanel painelPrincipal = new JPanel(new BorderLayout());
    painelPrincipal.add(painelN,BorderLayout.NORTH);
    painelPrincipal.add(painelC,BorderLayout.CENTER);
    painelPrincipal.add(new JPanel(),BorderLayout.EAST);
    painelPrincipal.add(new JPanel(),BorderLayout.WEST);
    painelPrincipal.add(painelS,BorderLayout.SOUTH);

    setContentPane(painelPrincipal);


    //Acções dos butões e campos de texto

    bOutraS.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        bOutraS.setEnabled(false);
        bConfirmar.setEnabled(false);
        lSolucao2.setText("Pesquisando Outra Solução...");

AgendaC.cliente.msgKQML("ask",AgendaC.cliente.idC,AgendaC.cliente.idS,"(other)"
);
      }
    });

    bConfirmar.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {

AgendaC.cliente.msgKQML("perform",AgendaC.cliente.idC,AgendaC.cliente.idS,"(con
firmMeeting(" + mensagem+ ',' + d + "))");
        eu.dispose();
      }
    });

    bCancelar.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {

AgendaC.cliente.msgKQML("perform",AgendaC.cliente.idC,AgendaC.cliente.idS,"(can
celMeeting)");
        eu.dispose();
      }
    });

    //para fechar o program ao carregar na cruz
    addWindowListener(new WindowAdapter()
    {
      public void windowClosing(WindowEvent e)
      {
        System.exit(0);
      }
    });

    marcacao(data, lista, c);
  }
```

```
  public void marcacao(String data, Object[] lista, float c)
  {
    String util = new String();

    for(int i = 0; i<lista.length; i++)
    {
      util = util + ',' + lista[i];
    }

    AgendaC.cliente.marcaReuniao(data+ ',' + c + util);
  }
}
```

JAcerca.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

//A janela acerca

class JAcerca extends JDialog
{
  //nome da janela principal
  private static final String titulo = "Acerca da Agenda 2000...";

  //construtor
  public JAcerca(JFrame janela)
  {
    super(janela, true);

    setTitle(titulo);
    setResizable(false);

    //saida
    addWindowListener(new WindowAdapter ()
    {
      public void windowClosing(WindowEvent e)
      {
        dispose();
      }
    });

    setSize(750, 525);
    getContentPane().add("Center", new imagem());
  }
}

class imagem extends Canvas
{
  Image img;

  public void paint(Graphics g)
  {
    img = Toolkit.getDefaultToolkit().getImage("imagens/acerca.jpg");

    //Desenha a imagem com escalamento
    g.drawImage(img, 0, 0, 750, 525, this);
  }
}
```

AlgC.java

```
import java.util.*;
import java.io.*;

//Classe de Algoritmos

class AlgC
{
  //constantes
  public static int HoraInicial = 7;
  public static int HoraFinal = 20;
  public static int TamLinhas = 390;

  //variaveis usadas por várias classes
  public static Hashtable horario;
  public static Ocup conflito;
  public static Ocup seleccao=null;
  public static int colunaSeleccao;

  //calendário para saber qual é a semana em causa
  public static Calendar agora = new GregorianCalendar();

  //variaveis para saber se a semana necessita de ser carregada do servidor
  public static int maxCache=0;
  public static int minCache=0;
  public static int semCache=0;

  //variável para esperar pela resposta
  public static int resposta = -1;

  //janela das comunicações
  public static JComunic comunic;


  public static Object[] ordenaListas(Object[] l1, Object[] l2)
  {
    int i, i1=0,i2=0, dif;
    int tam = l1.length + l2.length;
    Object[] lf = new Object[tam];
    String item1,item2;

    for(i=0;i<tam;i++)
    {
      if(i2==l2.length)
      {
        while(i1!=l1.length)
        {
          lf[i] = l1[i1];
          i1++;
          i++;
        }
        break;
      }
      else
      {
        if(i1==l1.length)
        {
          while(i2!=l2.length)
          {
            lf[i] = l2[i2];
            i2++;
            i++;
          }
          break;
        }
```

```
    }

    item1 = (String) l1[i1];
    item2 = (String) l2[i2];
    dif = item1.compareToIgnoreCase(item2);

    if(dif<0)
    {
      lf[i] = item1;
      i1++;
    }
    else
    {
      lf[i] = item2;
      i2++;
    }
  }
  return lf;
}

public static Object[] subtraiListas(Object[] l1, Object[] l2)
{
  int i,j,k=0;
  boolean n=true;
  int tam;
  String item1, item2;
  Object[] lf;

  if((tam=(l1.length - l2.length))<0)
    tam=-tam;

  lf = new Object[tam];

  for(i=0;i<l1.length;i++)
  {
    item1 = (String) l1[i];
    for(j=0;j<l2.length;j++)
    {
      item2 = (String) l2[j];
      if(item1.equals(item2))
      {
        n = false;
        break;
      }
    }
    if(n)
    {
     lf[k]=l1[i];
     k++;
    }
    n=true;
  }
  return lf;
}

public static boolean verificaData(int dia, int mes, int ano)
{

  if(dia<1 || dia >31)
    return false;

  if(mes<1 || mes >12)
    return false;

  if(ano<1999 || ano>2010)
```

```
      return false;

    return true;
  }

  public static boolean verificaDatas(int diai, int mesi, int anoi,int diaf,
int mesf, int anof)
  {
    if(anoi>anof)
      return false;

    if(anoi==anof && mesi>mesf)
      return false;

    if(anoi==anof && mesi==mesf && diai>diaf)
      return false;

    return true;
  }

  public static int passaParaHoras(int linha)
  {
    return (linha+HoraInicial);
  }

  public static void inicializaHorario()
  {
    horario = new Hashtable();
    AgendaC.cliente.pedeSemana(0);
  }

  public static Integer determinaChave(int dia, int mes, int ano)
  {
    return new Integer((dia*1000000 + mes*10000 + ano));
  }

  public static void insereOcupacao(int dia, int mes, int ano, Ocup ocupacao)
  {
    boolean flag = true;
    int hi, hiAux;
    LinkedList lista;
    ListIterator iLista;
    Ocup ocupacaoAux;
    Integer chave = determinaChave(dia,mes,ano);

    hi = ocupacao.devolveHInicial();
    lista = (LinkedList)horario.get(chave);

    if(lista==null)
    {
      lista = new LinkedList();
      lista.add(ocupacao);
    }
    else
    {
      iLista = lista.listIterator(0);
      for(int j=0;j<lista.size();j++)
      {
        ocupacaoAux = (Ocup)iLista.next();

        hiAux = ocupacaoAux.devolveHInicial();

        if(hi<hiAux)
        {
          iLista.previous();
```

```
          iLista.add(ocupacao);
          flag = false;
          break;
        }
      }

      if(flag)
        iLista.add(ocupacao);
    }
    horario.put(chave,lista);
  }

  public static void apagaOcupacao(int dia, int mes, int ano, Ocup ocupacao)
  {
    LinkedList lista;
    Integer chave = determinaChave(dia,mes,ano);

    lista = (LinkedList)horario.get(chave);

    if(lista.size()<=1)
    {
      horario.remove(chave);
    }
    else
    {
      lista.remove(ocupacao);
      horario.put(chave,lista);
    }

    seleccao=null;
    JAgenda.scrollpane.repaint();
  }

  public static boolean testaSeleccao(int dia, int mes, int ano, int hi, int
hf)
  {
    int hiAux, hfAux;
    LinkedList lista;
    ListIterator iLista;
    Ocup ocupacaoAux;
    Integer chave = determinaChave(dia,mes,ano);

    lista = (LinkedList)horario.get(chave);

    if(lista!=null)
    {
      iLista = lista.listIterator(0);
      for(int j=0;j<lista.size();j++)
      {
        ocupacaoAux = (Ocup)iLista.next();

        hiAux = ocupacaoAux.devolveHInicial();
        hfAux = ocupacaoAux.devolveHFinal();

        if(!(hf<=hiAux || hi>=hfAux))
        {
          conflito = ocupacaoAux;
          return false;
        }
      }
    }
    return true;
  }

  public synchronized static void mostra()
```

```
{
  int d,m,a,dia,mes,ano;
  List lista;
  ListIterator iLista;
  Ocup ocupacao;
  Integer chave;

  d = agora.get(agora.DAY_OF_MONTH);
  m = agora.get(agora.MONTH);
  a = agora.get(agora.YEAR);

  Calendar cal = new GregorianCalendar(a,m,d);

  cal.add(cal.DATE,-7);

  for(int i=1;i<=7;i++)
  {
    cal.add(cal.DATE,+1);

    dia = cal.get(cal.DAY_OF_MONTH);
    mes = cal.get(cal.MONTH)+1;
    ano = cal.get(cal.YEAR);

    chave = determinaChave(dia,mes,ano);

    lista = (LinkedList)horario.get(chave);

    if(lista!=null)
    {
      synchronized(lista)
      {
        iLista = lista.listIterator(0);

        while(iLista.hasNext())
        {
          ocupacao = (Ocup)iLista.next();
          System.out.println("Descricao: " + ocupacao.devolveDescricao());
          System.out.println("HoraI: " + ocupacao.devolveHInicial());
          System.out.println("HoraF: " + ocupacao.devolveHFinal());
          System.out.println("Prior: " + ocupacao.devolvePrioridade());
          System.out.println("");
        }
      }
    }
  }
}

public static int[] dataColuna(int coluna)
{
  int[] data = {0,0,0};
  //retira a coluna das horas;
  coluna--;

  //calculo do dia seleccionado

  agora.add(agora.DATE,-(6-coluna));
  data[0] = agora.get(agora.DAY_OF_MONTH);
  data[1] = agora.get(agora.MONTH)+1;
  data[2] = agora.get(agora.YEAR);
  agora.add(agora.DATE,+(6-coluna));

  return data;
}

public static int[] horasLinha(int[] linhas)
```

```
  {
    int linhai,linhaf;
    int[] hora = {0,0};

    if(linhas[0]<linhas[linhas.length-1])
    {
      linhai = linhas[0];
      linhaf = linhas[linhas.length-1]+1;
    }
    else
    {
      linhaf = linhas[0]+1;
      linhai = linhas[linhas.length-1];
    }

    hora[0] = passaParaHoras(linhai);
    hora[1] = passaParaHoras(linhaf);

    return hora;
  }

  public static int esperaResposta()
  {
    while(resposta == -1)
    {
      try
      {
        Thread.sleep(200);
      }
      catch(InterruptedException i)
      {
      System.err.println(i);
      }
    }
    int respostaAux = resposta;
    resposta = -1;

    return respostaAux;
  }
  public static String proximoValor(String str)
  {
    if(str.indexOf(",")!=-1)
      return str.substring(0,str.indexOf(","));
    else
      return str.substring(0,str.length()-2);
  }

  public static String retiraValor(String str)
  {
    return str.substring(str.indexOf(",")+1,str.length());
  }
}
```

Servidor:

AgendaS.java

```
public class AgendaS
{
  public static AgenteS servidor;
  public static AgenteR router;
```

```
  public static void main(String args[])
  {
    servidor = new AgenteS();
    servidor.start();
    router = new AgenteR();
    router.start();
  }
}
```

AgenteS.java

```
import BaseLayer.*;
import Abstract.*;
import KQMLLayer.*;
import java.lang.*;
import java.net.*;
import java.util.*;

import java.net.*;
import java.io.*;

public class AgenteS extends KQMLAgentAction
{
  public static String idC;
  public static String localC;
  public static String portoC;
  public static String tipoC;

  public static String idS;
  public static String localS;
  public static String portoS;
  public static String tipoS;

  private String login;
  private String password;

  public AgenteS()
  {
    super();

    String readline;

    //inicializa semana
    int diaSemana = AlgS.agora.get(Calendar.DAY_OF_WEEK);

    if(diaSemana==1)
      diaSemana = 8;

    AlgS.agora.add(AlgS.agora.DATE,+(8-diaSemana));

    //inicializa horário
    if(!AlgS.ler() || AlgS.horario==null)
      AlgS.inicializaHorario(); //inicializa o horário se não conseguir ler o
ficheiro


    try
    {
      // procedimentos de arranque
      _addresses = new BAddressTable();
      _connections = new BConnectionTable();
      _queue = new BMessageBuffer();
```

```
      // lê o ficheiro de endereco e coloca os seu dados numa tabela
      BufferedReader input = new BufferedReader(new FileReader(new
File("AgenteS")));

      //dados deste servidor
      while((readline = input.readLine()) != null)
      {
        if(!readline.startsWith("#"))
        {
          login = readline;
          break;
        }
      }

      while((readline = input.readLine()) != null)
      {
        if(!readline.startsWith("#"))
        {
          password = readline;
          break;
        }
      }

      while((readline = input.readLine()) != null)
      {
        if(!readline.startsWith("#"))
        {
          portoS = readline;
          break;
        }
      }

      InetAddress localhost = InetAddress.getLocalHost();
      System.out.println("Localização actual: " + localhost.getHostName());

      idS = "Servidor";
      localS = localhost.getHostName();
      tipoS = "AgenteS";

      System.out.println("Servidor: " + idS + ',' + localS + ',' + portoS + ','
+ tipoS);
      Address addr = new Address(idS + ',' + localS + ',' + portoS + ',' +
tipoS);
      _addresses.addAddress(addr);
      idS=addr.getID();

      input.close();

      // deve ser instanciado depois da tabela ter sido criada
      _security = new KQMLSecurity(_addresses);

      ServerThread server = createServerThread(idS,Thread.NORM_PRIORITY);
      if(server == null)
      {
        String args[] = new String[1];
        args[0] = idS;
        processMessage("InitializeFailed",args);
      }
    }
    catch (Exception e)
    {
      System.out.println(e.toString());
    }
  }
```

```
  public void processMessage(String command,Object obj)
  {
    String[] args = (String[])obj;

    if(command.equals("UnsupportedType"))
    {
      System.out.println(command + " " + args[0]);
    }
    else
    {
      if(command.equals("InitializeFailed"))
      {
        System.out.println(command + " " + args[0]);
        endAction();
      }
    }
  }

  protected boolean Act(Object o)
  {
    System.out.println((String) o);

    try
    {
      KQMLmessage mKQML = new KQMLmessage((String) o);

      String tipo = mKQML.getValue("performative");
      String emissor = mKQML.getValue("sender");
      String conteudo = mKQML.getValue("content");

      if(tipo.equals("perform") && conteudo.startsWith("(newConnection("))
        novaLigacaoCliente(conteudo.substring(15));
      else if(emissor.equals(idC))
           {
             if(tipo.equals("ask") && conteudo.startsWith("(week("))
               devolveSemana(conteudo.substring(6));
             else if(tipo.equals("ask") && conteudo.startsWith("(newMeeting("))
               novaMarcacao(conteudo.substring(12));
             else if(tipo.equals("tell") &&
conteudo.startsWith("(newOcupation("))
               novaOcupacao(conteudo.substring(14));
             else if(tipo.equals("tell") &&
conteudo.startsWith("(deleteOcupation("))
               apagaOcupacao(conteudo.substring(17));
             else if(tipo.equals("ask") && conteudo.startsWith("(other)"))
               outraSolucao();
             else if(tipo.equals("perform") &&
conteudo.startsWith("(confirmMeeting("))
               confirmarSolucao(conteudo.substring(16));
             else if(tipo.equals("perform") &&
conteudo.startsWith("(cancelMeeting)"))
               cancelarReuniao();
             else if(tipo.equals("ask") &&
conteudo.startsWith("(connectedUsers)"))
               devolveUtilizadores();
             else if(tipo.equals("ask-if") && conteudo.startsWith("(login("))
               verificaLogin(conteudo.substring(7));
             else if(tipo.equals("tell") && conteudo.startsWith("(bye)"))
               logoutCliente();
           }

      return true;
    }
    catch(KQMLLayer.ParseException e)
    {
```

```
      System.out.println(e.toString());
      return false;
    }
  }

  public boolean msgKQML(String tipo,String emissor,String receptor,String msg)
  {
    //criação de uma mensagem KQML
    KQMLmessage mKQML = new KQMLmessage();
    mKQML.addFieldValuePair("performative",tipo);
    mKQML.addFieldValuePair("sender",emissor);
    mKQML.addFieldValuePair("receiver",receptor);
    mKQML.addFieldValuePair("content",msg);

    //envio da mensagem
    try
    {
      sendMessage(mKQML);
      return true;
    }
    catch (ConnectionException e)
    {
      System.out.println(e.toString());
    }
    return false;
  }

  public void novaLigacaoCliente(String conteudo)
  {
    idC = AlgS.proximoValor(conteudo);
    conteudo = AlgS.retiraValor(conteudo);
    localC = AlgS.proximoValor(conteudo);
    conteudo = AlgS.retiraValor(conteudo);
    portoC = AlgS.proximoValor(conteudo);
    conteudo = AlgS.retiraValor(conteudo);
    tipoC = AlgS.proximoValor(conteudo);

    try
    {
      Address addr = new Address(idC + ',' + localC + ',' + portoC + ',' +
tipoC);
      _addresses.addAddress(addr);
    }
    catch (Exception e)
    {
      System.out.println(e.toString());
    }
  }

  public boolean verificaLogin(String conteudo)
  {
    String l = AlgS.proximoValor(conteudo);
    conteudo = AlgS.retiraValor(conteudo);
    String p = AlgS.proximoValor(conteudo);

    if(l.equals(login) && p.equals(password))
    {
      if(msgKQML("reply",idS,idC,"(login(true))"))
        return true;
    }
    else
      if(msgKQML("reply",idS,idC,"(login(false))"))
        return true;

    return false;
```

```
}

public void novaOcupacao(String conteudo)
{
  int dia = Integer.parseInt(AlgS.proximoValor(conteudo));
  conteudo = AlgS.retiraValor(conteudo);
  int mes = Integer.parseInt(AlgS.proximoValor(conteudo));
  conteudo = AlgS.retiraValor(conteudo);
  int ano = Integer.parseInt(AlgS.proximoValor(conteudo));
  conteudo = AlgS.retiraValor(conteudo);
  String desc = AlgS.proximoValor(conteudo);
  conteudo = AlgS.retiraValor(conteudo);
  int horai = Integer.parseInt(AlgS.proximoValor(conteudo));
  conteudo = AlgS.retiraValor(conteudo);
  int horaf = Integer.parseInt(AlgS.proximoValor(conteudo));
  conteudo = AlgS.retiraValor(conteudo);
  int prior = Integer.parseInt(AlgS.proximoValor(conteudo));

  Ocup ocupacao = new Ocup(desc,horai,horaf,prior);
  AlgS.insereOcupacao(dia,mes,ano,ocupacao);
}

public void apagaOcupacao(String conteudo)
{
  int dia = Integer.parseInt(AlgS.proximoValor(conteudo));
  conteudo = AlgS.retiraValor(conteudo);
  int mes = Integer.parseInt(AlgS.proximoValor(conteudo));
  conteudo = AlgS.retiraValor(conteudo);
  int ano = Integer.parseInt(AlgS.proximoValor(conteudo));
  conteudo = AlgS.retiraValor(conteudo);
  String desc = AlgS.proximoValor(conteudo);
  conteudo = AlgS.retiraValor(conteudo);
  int horai = Integer.parseInt(AlgS.proximoValor(conteudo));
  conteudo = AlgS.retiraValor(conteudo);
  int horaf = Integer.parseInt(AlgS.proximoValor(conteudo));
  conteudo = AlgS.retiraValor(conteudo);
  int prior = Integer.parseInt(AlgS.proximoValor(conteudo));

  AlgS.apagaOcupacao(dia,mes,ano,horai);
  //AlgS.gravar();
}

public boolean devolveSemana(String conteudo)
{
  int sem = Integer.parseInt(AlgS.proximoValor(conteudo));

  AlgS.agora = new GregorianCalendar();

  //inicializa semana
  int diaSemana = AlgS.agora.get(Calendar.DAY_OF_WEEK);

  if(diaSemana==1)
    diaSemana = 8;

  AlgS.agora.add(AlgS.agora.DATE,+(8-diaSemana));

  AlgS.agora.add(AlgS.agora.DATE,+(sem*7));

  int d = AlgS.agora.get(AlgS.agora.DAY_OF_MONTH);
  int m = AlgS.agora.get(AlgS.agora.MONTH)+1;
  int a = AlgS.agora.get(AlgS.agora.YEAR);

  if(msgKQML("reply",idS,idC,"(week("+AlgS.semanaHorario()+"))"))
    return true;
  else return false;
```

```
}

public void logoutCliente()
{
   AlgS.gravar();
   _connections.removeConnection(idC);
}

public void devolveUtilizadores()
{
  String util;
  util = AgendaS.router.listaUtilizadores();
  msgKQML("reply",idS,idC,"(connectedUsers("+ util +"))");
}

public synchronized void novaMarcacao(String conteudo)
{
  String data, hora, duracao, aux;
  Ocup ocupacao;
  Integer chave;
  ListIterator iLista;
  int prioridade, hi, hf, n=0;

  AlgS.fim = -1;
  AlgS.auxFim = 0;
  AlgS.auxSol = 0;

  data = AlgS.proximoValor(conteudo);
  conteudo = AlgS.retiraValor(conteudo);
  hora = AlgS.proximoValor(conteudo);
  conteudo = AlgS.retiraValor(conteudo);
  duracao = AlgS.proximoValor(conteudo);
  conteudo = AlgS.retiraValor(conteudo);
  AlgS.media = Float.parseFloat(AlgS.proximoValor(conteudo));
  conteudo = AlgS.retiraValor(conteudo);

  aux = conteudo;

  while(aux.indexOf(",")!=-1)
  {
    aux = AlgS.retiraValor(aux);
    n++;
  }

  String[] util = new String[n+1];

  n = 0;

  while(conteudo.indexOf(",")!=-1)
  {
    util[n] = AlgS.proximoValor(conteudo);
    conteudo = AlgS.retiraValor(conteudo);
    n++;
  }
  util[n] = AlgS.proximoValor(conteudo);

  AlgS.nomeR = new String[util.length];
  AlgS.nomeR = util;

  AlgS.solucaoR = new Integer[(util.length+1)][14];
  for(int i=0; i<util.length; i++)
    for(int j=0; j<14; j++)
      AlgS.solucaoR[i][j] = new Integer(-1);
```

```
    chave = new Integer(data);

    //inicializa o meu array
    for(int i=0; i<14; i++)
      AlgS.solucaoR[util.length][i] = new Integer(0);

    List lista = (List)AlgS.horario.get(chave);

    if(lista!=null)
    {
      synchronized(lista)
      {
        iLista = lista.listIterator(0);

        while(iLista.hasNext())
        {
          ocupacao = (Ocup)iLista.next();
          hi = ocupacao.devolveHInicial()-7;
          hf = ocupacao.devolveHFinal()-7;
          prioridade = ocupacao.devolvePrioridade();

          while(hi!=hf)
          {
            AlgS.solucaoR[util.length][hi] = new Integer(prioridade);
            hi++;
          }
        }
      }
    }

    //inicializa média
    for(int i=0; i<14; i++)
    {
      AlgS.mediaR[i] = new
Float((AlgS.solucaoR[util.length][i].floatValue()/(float)(util.length+1)));
      //System.out.print("" + AlgS.mediaR[i].floatValue() +  " , ");
    }

    for(int i=0; i<util.length; i++)
    {
      AgendaS.router.msgKQML("ask",AgendaS.router.idS,
util[i],"(newMeetingDate(" + data + ',' + duracao + "))");
      msgKQML("perform",idS,idC,"(stdout1(" + util[i] + ",newMeetingDate(" +
data + ',' + duracao + ")))");
    }

    for(int i=0; i<util.length; i++)
    {
      AgendaS.router.msgKQML("ask",AgendaS.router.idS,util[i],"(meeting(" +
hora + "))");
      msgKQML("perform",idS,idC,"(stdout1(" + util[i] + ",meeting(" + hora +
")))");
    }
  }

  public void outraSolucao()
  {
    int novaHora, n;
    Float x;
    boolean flag = true;

    AlgS.mediaR[AlgS.solucao] = new Float(6);

    for(int i=0; i<AlgS.mediaR.length; i++)
    {
```

```
      n=0;

      for(int j=0; j<AlgS.nomeR.length+1; j++)
        if(AlgS.solucaoR[j][i].intValue()!=-1)
          n++;

      if((AlgS.nomeR.length+1==n) &&
(AlgS.mediaS=AlgS.mediaR[i].floatValue())<=AlgS.media)
      {
        AlgS.solucao = i;

        System.out.println("CHEGUEI AO FIM");

AgendaS.servidor.msgKQML("perform",AgendaS.servidor.idS,AgendaS.servidor.idC,"(
solution(" + (AlgS.solucao+7) + ',' + AlgS.mediaS + "))");
        flag = false;
        break;
      }
    }

    if(flag)
    {
      AlgS.fim = -1;
      AlgS.auxSol = 0;

      //acha o valor não ocupado com a menor média
      for(int j=0; j<AlgS.nomeR.length; j++)
      {
        x = new Float(6);
        novaHora = -1;

        for(int i=0; i<14; i++)
          if(AlgS.solucaoR[j][i].intValue()==-1)
            if(AlgS.mediaR[i].floatValue()<x.floatValue())
            {
              x = AlgS.mediaR[i];
              novaHora = i+7;
              System.out.print("Nova Hora: " + novaHora + " , ");
            }

        if(AlgS.fim == -1 && novaHora!=-1)
        {

AgendaS.router.msgKQML("ask",AgendaS.router.idS,AlgS.nomeR[j],"(meeting(" +
novaHora + "))");
          msgKQML("perform",idS,idC,"(stdout1(" + AlgS.nomeR[j] + ",meeting(" +
novaHora + ")))");
        }
        else
          if(novaHora == -1)
          {
            AlgS.auxFim++;
            if(AlgS.auxFim == AlgS.nomeR.length)
            {
              System.out.println("ESGOTEI AS SOLUCOES");
              msgKQML("perform",idS,idC,"(solution(-1))");
            }
          }
      }
    }
  }

  public void confirmarSolucao(String conteudo)
  {
    for(int i=0; i<AlgS.nomeR.length; i++)
```

```
    {

AgendaS.router.msgKQML("perform",AgendaS.router.idS,AlgS.nomeR[i],"(confirmedMe
eting(" + (AlgS.solucao+7) + ',' + conteudo);
    }

    int hora = (AlgS.solucao+7);

    String desc = AlgS.proximoValor(conteudo);
    conteudo = AlgS.retiraValor(conteudo);
    Integer chave = new Integer(AlgS.proximoValor(conteudo));
    conteudo = AlgS.retiraValor(conteudo);
    conteudo = AlgS.retiraValor(conteudo);
    int duracao = Integer.parseInt(AlgS.proximoValor(conteudo));

    Ocup ocupacao = new Ocup(desc,hora,(hora+duracao),3);

    AlgS.insereReuniao(chave, ocupacao);
  }

  public void cancelarReuniao()
  {
    AlgS.fim = 1;
  }
}
```

AgenteR.java

```
import Abstract.*;
import KQMLLayer.*;
import RouterLayer.AgentClient.*;

import java.io.*;
import java.util.*;
import java.net.*;

public class AgenteR extends RouterClientAction
{
  public static String idS;
  public static String localS;
  public static String portoS;
  public static String tipoS;
  public static String descricaoS;

  public static int sincro = 0;

  public static int listaResp = 0;
  public static String utilizadores;

  public AgenteR()
  {
    super();

    Address myAddress ,router, registrar;
    String readline;

    try
    {

      // lê o ficheiro de enderecos e coloca os seu dados numa tabela
      BufferedReader input = new BufferedReader(new FileReader(new
File("AgenteS")));
```

```
      //dados deste servidor
      while((readline = input.readLine()) != null)
      {
        if(!readline.startsWith("#"))
        {
          idS = readline;

          while((readline = input.readLine()) != null)
          {
            if(!readline.startsWith("#"))
            {
              break;
            }
          }

          while((readline = input.readLine()) != null)
          {
            if(!readline.startsWith("#"))
            {
              break;
            }
          }

          while((readline = input.readLine()) != null)
          {
            if(!readline.startsWith("#"))
            {
              portoS = readline;
              break;
            }
          }

          while((readline = input.readLine()) != null)
          {
            if(!readline.startsWith("#"))
            {
              descricaoS = readline;
              break;
            }
          }

          InetAddress localhost = InetAddress.getLocalHost();
          System.out.println("Localização actual: " + localhost.getHostName());

          localS = localhost.getHostName();
          tipoS = "AgenteS";

          System.out.println("Servidor: " + idS + ',' + localS + ',' + portoS +
',' + tipoS + ',' + descricaoS);
          myAddress = new Address(idS + ',' + localS + ',' + portoS + ',' +
tipoS + ',' + descricaoS);

          setMyAddress(myAddress);
          idS = myAddress.getID();

          break;
        }
      }

      while((readline = input.readLine()) != null)
      {
        if(!readline.startsWith("#"))
        {
          System.out.println("Router: " + readline);
          router = new Address(readline);
```

```
          setRouterAddress(router);

          break;
        }
      }

      while((readline = input.readLine()) != null)
      {
        if(!readline.startsWith("#"))
        {
          System.out.println("Registrar: " + readline);
          registrar = new Address(readline);
          setRegistrarAddress(registrar);

          break;
        }
      }
    }
    catch (Exception e)
    {
      System.out.println(e.toString());
      System.exit(1);
    }

    try
    {
      createServerThread(idS,Thread.NORM_PRIORITY);
      register();
      connect();
    }
    catch(ConnectionException e)
    {
      System.out.println(e.toString());
      System.exit(1);
    }
  }

  public boolean Act(Object o)
  {
    try
    {
      KQMLmail mail = new KQMLmail((String) o, 0);
      _mailQueue.addElement(mail);
      addToDeleteBuffer(0);
      System.out.println((String) o);

      KQMLmessage mKQML = mail.getKQMLmessage();

      String tipo = mKQML.getValue("performative");
      String emissor = mKQML.getValue("sender");
      String conteudo = mKQML.getValue("content");

      if(emissor.equals("Router") && tipo.equals("registered-agent"))
      {
        utilizadores = conteudo.substring(1);
        AlgS.respostaRouter = 1;
      }
      else if(tipo.equals("ask") && conteudo.startsWith("(meeting("))
        devolvePreferencia(emissor, conteudo.substring(9));
      else if(tipo.equals("reply") && conteudo.startsWith("(meetingAnswer("))
        processaPreferencia(emissor, conteudo.substring(15));
      else if(tipo.equals("ask") && conteudo.startsWith("(newMeetingDate("))
        carregaLista(conteudo.substring(16));
      else if(tipo.equals("perform") &&
conteudo.startsWith("(confirmedMeeting("))
```

```
        confirmaReuniao(conteudo.substring(18));
      }
      catch(Exception e)
      {
        System.out.println(e.toString());
        return false;
      }

      return true;
    }

    public void processMessage(String command, Object obj)
    {

    }

    public boolean msgKQML(String tipo,String emissor,String receptor,String msg)
    {
      //criação de uma mensagem KQML
      KQMLmessage mKQML = new KQMLmessage();
      mKQML.addFieldValuePair("performative",tipo);
      mKQML.addFieldValuePair("sender",emissor);
      mKQML.addFieldValuePair("receiver",receptor);
      mKQML.addFieldValuePair("content",msg);

      //envio da mensagem
      try
      {
        sendMessage(mKQML);
        return true;
      }
      catch (ConnectionException e)
      {
        System.out.println(e.toString());
      }
      return false;
    }

    public String listaUtilizadores()
    {
      String arg, id, estado;
      String util = new String();

      try
      {
/*
        listUsers();
        AlgS.esperaRespostaRouter();

        do
        {
          utilizadores = AlgS.retiraArgumento(utilizadores);
          arg = AlgS.proximoArgumento(utilizadores);

          id = AlgS.proximoValorEspaco(arg);
          arg = AlgS.retiraValorEspaco(arg);
          arg = AlgS.retiraValorEspaco(arg);
          arg = AlgS.retiraValorEspaco(arg);
          estado = AlgS.proximoValorEspaco(arg);

          if(estado.equals("connected") && !id.equals(idS))
          {
            msgKQML("tell",idS,id,"(Hello)");
          }
        }while(utilizadores.indexOf('(')!=-1);
```

```
        try
        {
          Thread.sleep(500);
        }
        catch(InterruptedException f)
        {
        System.err.println(f);
        }
*/

      listUsers();
      AlgS.esperaRespostaRouter();

      do
      {
        utilizadores = AlgS.retiraArgumento(utilizadores);
        arg = AlgS.proximoArgumento(utilizadores);

        id = AlgS.proximoValorEspaco(arg);
        arg = AlgS.retiraValorEspaco(arg);
        arg = AlgS.retiraValorEspaco(arg);
        arg = AlgS.retiraValorEspaco(arg);
        estado = AlgS.proximoValorEspaco(arg);

        if(estado.equals("connected") && !id.equals(idS))
        {
          util = util + ',' + id;
        }
      }while(utilizadores.indexOf('(')!=-1);
    }
    catch (ConnectionException e)
    {
      System.out.println(e.toString());
    }

    if(util.length()>0)
      return util.substring(1);
    else
      return "";
  }

  public synchronized void carregaLista(String conteudo)
  {
    Ocup ocupacao;
    Integer chave;
    ListIterator iLista;
    int prioridade, hi, hf;

    sincro = -1;

    chave = new Integer(AlgS.proximoValor(conteudo));
    conteudo = AlgS.retiraValor(conteudo);
    AlgS.duracaoR = Integer.parseInt(AlgS.proximoValor(conteudo));

    //inicializa historial
    for(int i=0; i<14; i++)
      AlgS.historialR[i] = new Integer(0);

    List lista = (List)AlgS.horario.get(chave);

    if(lista!=null)
    {
      synchronized(lista)
      {
```

```
        iLista = lista.listIterator(0);

        while(iLista.hasNext())
        {
          ocupacao = (Ocup)iLista.next();
          hi = ocupacao.devolveHInicial()-7;
          hf = ocupacao.devolveHFinal()-7;
          prioridade = ocupacao.devolvePrioridade();

          while(hi!=hf)
          {
            AlgS.historialR[hi] = new Integer(prioridade);
            hi++;
          }
        }
      }
    }
    sincro=0;
  }

  public synchronized void devolvePreferencia(String receptor, String conteudo)
  {
    String prior;
    int altHora=-1, altPrior=-1, br=-1;
    int hora = Integer.parseInt(AlgS.proximoValor(conteudo));

    prior = AlgS.historialR[hora-7].toString();
    AlgS.historialR[hora-7] = new Integer(-1);

    for(int j=0; j<6; j++)
    {
      for(int i=0; i<14; i++)
        if(AlgS.historialR[i].intValue()==j)
        {
          altHora=i+7;
          altPrior=AlgS.historialR[i].intValue();
          AlgS.historialR[i] = new Integer(-1);
          br=0;
          break;
        }
      if(br==0)
        break;
    }

    while(sincro == -1)
    {
      try
      {
        Thread.sleep(200);
      }
      catch(InterruptedException i)
      {
      System.err.println(i);
      }
    }

    if(altHora!=-1)
      msgKQML("reply",idS,receptor,"(meetingAnswer(" + hora + ',' + prior + ','
+ "yes" + ',' + altHora + ',' + altPrior + "))");
    else
      msgKQML("reply",idS,receptor,"(meetingAnswer(" + hora + ',' + prior + ','
+ "no))");
  }

  public synchronized void processaPreferencia(String emissor, String conteudo)
```

```
  {
    int hora, prior, altHora, altPrior, nUtil=-1, novaHora=-1;
    String simNao;
    Float x = new Float(6);


AgendaS.servidor.msgKQML("perform",AgendaS.servidor.idS,AgendaS.servidor.idC,"(
stdout2(" + emissor + ',' + conteudo.substring(0,conteudo.length()-2) + "))");

    hora = Integer.parseInt(AlgS.proximoValor(conteudo))-7;
    conteudo = AlgS.retiraValor(conteudo);
    prior = Integer.parseInt(AlgS.proximoValor(conteudo));
    conteudo = AlgS.retiraValor(conteudo);
    simNao = AlgS.proximoValor(conteudo);

    //decobre qual o indice do utilizador que enviou a resposta
    for(int i=0; i<AlgS.nomeR.length; i++)
      if(AlgS.nomeR[i].equals(emissor))
        nUtil=i;

    //calcula a média
    AlgS.solucaoR[nUtil][hora] = new Integer(prior);
    if(AlgS.fim == -1 && AlgS.calculaMedia(hora, AlgS.nomeR.length)==1)
      AlgS.fim = 0;


    if(simNao.equals("yes"))
    {
      conteudo = AlgS.retiraValor(conteudo);
      altHora = Integer.parseInt(AlgS.proximoValor(conteudo))-7;
      conteudo = AlgS.retiraValor(conteudo);
      altPrior = Integer.parseInt(AlgS.proximoValor(conteudo));

      //calcula média para a hora alternativa
      AlgS.solucaoR[nUtil][altHora] = new Integer(altPrior);
      if(AlgS.fim == -1 && AlgS.calculaMedia(altHora, AlgS.nomeR.length)==1)
        AlgS.fim = 0;
    }

    if(AlgS.fim == -1)
    {
      //acha o valor não ocupado com a menor média
      for(int i=0; i<14; i++)
        if(AlgS.solucaoR[nUtil][i].intValue()==-1)
          if(AlgS.mediaR[i].floatValue()<x.floatValue())
          {
            x = AlgS.mediaR[i];
            novaHora = i+7;
            System.out.println("Nova Hora: " + novaHora);
          }
      if(AlgS.fim == -1 && novaHora!=-1)
      {
        msgKQML("ask",idS,AlgS.nomeR[nUtil],"(meeting(" + novaHora + "))");

AgendaS.servidor.msgKQML("perform",AgendaS.servidor.idS,AgendaS.servidor.idC,"(
stdout1(" + AlgS.nomeR[nUtil] + ",meeting(" + novaHora + ")))");
      }
    }
    else
      if(AlgS.fim == 0)
      {
        AlgS.auxSol++;
        if((AlgS.auxSol + AlgS.auxFim) == AlgS.nomeR.length)
        {
          System.out.println("CHEGUEI AO FIM");
```

```
AgendaS.servidor.msgKQML("perform",AgendaS.servidor.idS,AgendaS.servidor.idC,"(
solution(" + (AlgS.solucao+7) + ',' + AlgS.mediaS + "))");
        }
      }

    if(AlgS.fim == -1 && novaHora == -1)
    {
      AlgS.auxFim++;
      if(AlgS.auxFim == AlgS.nomeR.length)
      {
        System.out.println("ESGOTEI AS SOLUCOES");

AgendaS.servidor.msgKQML("perform",AgendaS.servidor.idS,AgendaS.servidor.idC,"(
solution(-1))");
      }
    }
  }

  public void confirmaReuniao(String conteudo)
  {
    int hora = Integer.parseInt(AlgS.proximoValor(conteudo));
    conteudo = AlgS.retiraValor(conteudo);
    String desc = AlgS.proximoValor(conteudo);
    conteudo = AlgS.retiraValor(conteudo);
    Integer chave = new Integer(AlgS.proximoValor(conteudo));
    conteudo = AlgS.retiraValor(conteudo);
    conteudo = AlgS.retiraValor(conteudo);
    int duracao = Integer.parseInt(AlgS.proximoValor(conteudo));

    Ocup ocupacao = new Ocup(desc,hora,(hora+duracao),3);

    AlgS.insereReuniao(chave, ocupacao);
  }
}
```

AlgS.java

```
import java.util.*;
import java.io.*;

//Classe de Algoritmos

class AlgS
{
  //constantes
  public static String nomeFicheiro = "horario.hor";
  public static float media = 1;
  public static int fim = -1;
  public static int auxFim = 0;
  public static int auxSol = 0;

  //variaveis usadas por várias classes
  public static Hashtable horario;
  public static Ocup conflito;

  //calendário para saber qual é a semana em causa
  public static Calendar agora = new GregorianCalendar();

  //variável para esperar pela resposta do router
  public static int respostaRouter = -1;
```

```
  //duração da reunião
  public static int duracaoR;

  //historiais
  public static Integer[] historialR = new Integer[14];
  public static String[] nomeR;
  public static Float[] mediaR = new Float[14];
  public static Integer[][] solucaoR;
  public static int solucao;
  public static float mediaS = 0;

  public static void inicializaHorario()
  {
    horario = new Hashtable();
  }

  public synchronized static Integer determinaChave(int dia, int mes, int ano)
  {
    return new Integer((dia*1000000 + mes*10000 + ano));
  }

  public synchronized static void insereOcupacao(int dia, int mes, int ano,
Ocup ocupacao)
  {
    boolean flag = true;
    int hi, hiAux;
    List lista;
    ListIterator iLista;
    Ocup ocupacaoAux;
    Integer chave = determinaChave(dia,mes,ano);

    hi = ocupacao.devolveHInicial();
    lista = (List)horario.get(chave);

    if(lista==null)
    {
      lista = Collections.synchronizedList(new LinkedList());
      lista.add(ocupacao);
    }
     else
     {
      synchronized(lista)
      {
        iLista = lista.listIterator(0);
        for(int j=0;j<lista.size();j++)
        {
          ocupacaoAux = (Ocup)iLista.next();

          hiAux = ocupacaoAux.devolveHInicial();

          if(hi<hiAux)
          {
            iLista.previous();
            iLista.add(ocupacao);
            flag = false;
            break;
          }
        }

        if(flag)
          iLista.add(ocupacao);
       }
     }
     horario.put(chave,lista);
   }
```

```
  public synchronized static void insereOcupacao(Integer chave, Ocup ocupacao)
  {
    boolean flag = true;
    int hi, hiAux;
    List lista;
    ListIterator iLista;
    Ocup ocupacaoAux;

    hi = ocupacao.devolveHInicial();
    lista = (List)horario.get(chave);

    if(lista==null)
    {
      lista = Collections.synchronizedList(new LinkedList());
      lista.add(ocupacao);
    }
     else
     {
      synchronized(lista)
      {
        iLista = lista.listIterator(0);
        for(int j=0;j<lista.size();j++)
        {
          ocupacaoAux = (Ocup)iLista.next();

          hiAux = ocupacaoAux.devolveHInicial();

          if(hi<hiAux)
          {
            iLista.previous();
            iLista.add(ocupacao);
            flag = false;
            break;
          }
        }

        if(flag)
          iLista.add(ocupacao);
       }
     }
     horario.put(chave,lista);
   }

  public synchronized static void apagaOcupacao(int dia, int mes, int ano, int
horai)
  {
    List lista;
    ListIterator iList;
    Ocup ocup;
    Integer chave = determinaChave(dia,mes,ano);

    lista = (List)horario.get(chave);

    if(lista.size()<=1)
    {
      horario.remove(chave);
    }
    else
    {
      synchronized(lista)
      {
        iList = lista.listIterator();
        while(iList.hasNext())
        {
```

```
          ocup = (Ocup)iList.next();
          if(horai==ocup.devolveHInicial())
          {
            lista.remove(ocup);
            break;
          }
        }
      }
      horario.put(chave,lista);
    }
  }

  public synchronized static String semanaHorario()
  {
    int d,m,a,dia,mes,ano;
    Integer chave;
    List lista;
    ListIterator iLista;
    Ocup ocupacao;
    String semana = new String("");

    d = agora.get(agora.DAY_OF_MONTH);
    m = agora.get(agora.MONTH);
    a = agora.get(agora.YEAR);

    Calendar cal = new GregorianCalendar(a,m,d);

    cal.add(cal.DATE,-7);

    for(int i=1;i<=7;i++)
    {
      cal.add(cal.DATE,+1);

      dia = cal.get(cal.DAY_OF_MONTH);
      mes = cal.get(cal.MONTH)+1;
      ano = cal.get(cal.YEAR);

      chave = determinaChave(dia,mes,ano);

      lista = (List)horario.get(chave);

      if(lista!=null)
      {
        synchronized(lista)
        {
          iLista = lista.listIterator(0);

          while(iLista.hasNext())
          {
            ocupacao = (Ocup)iLista.next();
            semana = semana+ dia + "," + mes + "," + ano + ",";
            semana =
semana+ocupacao.devolveDescricao()+","+ocupacao.devolveHInicial()+","+ocupacao.
devolveHFinal()+","+ocupacao.devolvePrioridade()+",";
          }
        }
      }
    }
    if(semana.length()>0)
      semana = semana.substring(0,(semana.length()-1));

    return semana;
  }

  public static void gravar()
```

```
{
  try
  {
    // Serializa os objectos para o ficheiro
    FileOutputStream out = new FileOutputStream(nomeFicheiro);
    ObjectOutputStream s = new ObjectOutputStream(out);

    s.writeObject(horario);
    s.flush();

    s.close();
    out.close();
  }
  catch(Exception e)
  {
    System.out.println(e.toString());
  }
}

public static boolean ler()
{
  try
  {

     FileInputStream in = new FileInputStream(nomeFicheiro);
     ObjectInputStream s = new ObjectInputStream(in);

     horario = (Hashtable) s.readObject();

    in.close();
    return true;
  }
  catch(Exception e)
  {
    System.out.println(e.toString());
    return false;
  }
}

public synchronized static void mostra()
{
  int d,m,a,dia,mes,ano;
  List lista;
  ListIterator iLista;
  Ocup ocupacao;
  Integer chave;

  d = agora.get(agora.DAY_OF_MONTH);
  m = agora.get(agora.MONTH);
  a = agora.get(agora.YEAR);

  Calendar cal = new GregorianCalendar(a,m,d);

  cal.add(cal.DATE,-7);

  for(int i=1;i<=7;i++)
  {
    cal.add(cal.DATE,+1);

    dia = cal.get(cal.DAY_OF_MONTH);
    mes = cal.get(cal.MONTH)+1;
    ano = cal.get(cal.YEAR);

    chave = determinaChave(dia,mes,ano);
```

```
    lista = (List)horario.get(chave);

    if(!(lista==null))
    {
      synchronized(lista)
      {
        iLista = lista.listIterator(0);

        while(iLista.hasNext())
        {
          ocupacao = (Ocup)iLista.next();
          System.out.println("Descricao: " + ocupacao.devolveDescricao());
          System.out.println("HoraI: " + ocupacao.devolveHInicial());
          System.out.println("HoraF: " + ocupacao.devolveHFinal());
          System.out.println("Prior: " + ocupacao.devolvePrioridade());
          System.out.println("");
        }
      }
    }
  }
}

public static int esperaRespostaRouter()
{
  while(respostaRouter == -1)
  {
    try
    {
      Thread.sleep(200);
    }
    catch(InterruptedException i)
    {
    System.err.println(i);
    }
  }
  int respostaAux = respostaRouter;
  respostaRouter = -1;

  return respostaAux;
}

public synchronized static int calculaMedia(int hora, int nUtil)
{
  int soma = 0, n=0;

  for(int i=0; i<nUtil; i++)
    if(solucaoR[i][hora].intValue()!=-1)
    {
      soma = soma + solucaoR[i][hora].intValue();
      n++;
    }

  soma = soma + solucaoR[nUtil][hora].intValue();
  n++;

  mediaR[hora] = new Float((float)soma/(float)n);

  System.out.println("Hora: " + (hora+7) + "  Média: " + mediaR[hora]);

  if((nUtil+1)==n && (mediaS=mediaR[hora].floatValue())<=media)
  {
    solucao = hora;
    return 1;
  }
  else return 0;
```

```
}

public synchronized static void insereReuniao(Integer chave, Ocup ocupacao)
{
  int hi, hf, hiAux, hfAux;
  List lista;
  ListIterator iLista;
  Ocup ocupacaoAux;

  hi = ocupacao.devolveHInicial();
  hf = ocupacao.devolveHFinal();

  lista = (List)horario.get(chave);

  if(lista!=null)
  {
    synchronized(lista)
    {
      iLista = lista.listIterator(0);
      for(int j=0;j<lista.size();j++)
      {
        ocupacaoAux = (Ocup)iLista.next();

        hiAux = ocupacaoAux.devolveHInicial();
        hfAux = ocupacaoAux.devolveHFinal();

        if((hfAux>hi && hfAux<=hf) || (hiAux>=hi && hiAux<hf))
        {
          lista.remove(ocupacaoAux);
        }
      }
    }
  }
  horario.put(chave,lista);

  insereOcupacao(chave, ocupacao);
}

public synchronized static String proximoValor(String str)
{
  if(str.indexOf(",")!=-1)
    return str.substring(0,str.indexOf(","));
  else
    return str.substring(0,str.length()-2);
}

public synchronized static String retiraValor(String str)
{
  return str.substring(str.indexOf(",")+1,str.length());
}

public synchronized static String proximoValorEspaco(String str)
{
  if(str.indexOf(' ')!=-1)
    return str.substring(0,str.indexOf(' '));
  else
    return str.substring(0,str.length());
}

public synchronized static String retiraValorEspaco(String str)
{
  return str.substring(str.indexOf(' ')+1,str.length());
}

public synchronized static String proximoArgumento(String str)
```

```
  {
    return str.substring(0,str.indexOf(')'));
  }

  public synchronized static String retiraArgumento(String str)
  {
    return str.substring(str.indexOf('(')+1,str.length());
  }
}
```